



UnboundID[®] Identity Proxy Administration Guide

Version 5.0.1

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

Copyright

This document constitutes an unpublished, copyrighted work and contains valuable trade secrets and other confidential information belonging to UnboundID Corporation. None of the foregoing material may be copied, duplicated, or disclosed to third parties without the express written permission of UnboundID Corporation.

This distribution may include materials developed by third parties. Third-party URLs are also referenced in this document. UnboundID is not responsible for the availability of third-party web sites mentioned in this document. UnboundID does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. UnboundID will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

“UnboundID” is a registered trademark of UnboundID Corporation. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

The contents of this publication are presented for information purposes only and is provided “as is”. While every effort has been made to ensure the accuracy of the contents, the contents are not to be construed as warranties or guarantees, expressed or implied, regarding the products or services described herein or their use or applicability. We reserve the right to modify or improve the design or specifications of such products at any time without notice.

Copyright 2015 UnboundID Corporation

All Rights Reserved

Published: 2015-04-08

Contents

Preface.....	xi
Purpose of This Guide.....	xi
Audience.....	xi
Related Documentation.....	xi
Document Conventions.....	xii
 Chapter 1: Introduction.....	 1
Overview of the UnboundID Identity Proxy Features.....	2
Overview of the Identity Proxy Components and Terminology.....	3
About Locations.....	3
About LDAP External Servers.....	4
About LDAP Health Checks.....	4
About Load-Balancing Algorithms.....	6
About Proxy Transformations.....	7
About Request Processors.....	7
About Server Affinity Providers.....	8
About Subtree Views.....	8
About the Connection Pools.....	8
About Client Connection Policies.....	9
About Entry Balancing.....	9
Server Component Architecture.....	10
Architecture of a Simple Identity Proxy Deployment.....	10
Architecture of an Entry-Balancing Identity Proxy Deployment.....	10
Identity Proxy Configuration Overview.....	12
 Chapter 2: Installing the Identity Proxy.....	 13
Before You Begin.....	14
Defining a Naming Strategy for Server Locations.....	14
Software Requirements: Java.....	14
Preparing the Operating System.....	15
Configuring the File Descriptor Limits.....	15
To Set the Filesystem Flushes.....	15
About Editing OS-Level Environment Variables.....	16
Install sysstat and pstack (Red Hat).....	16
Install dstat (SUSE Linux).....	16
Omit vm.overcommit_memory.....	16
Getting the Installation Packages.....	17
To Unpack the Build Distribution.....	17
About the RPM Package.....	17
Installing the Identity Proxy.....	18
About the setup Tool.....	18
Installing the First Identity Proxy in Interactive Mode.....	18
Installing the First Identity Proxy in Non-Interactive Mode.....	24
To Install Additional Identity Proxy in Non-Interactive Mode.....	25
Installing the Identity Proxy with a Truststore in Non-Interactive Mode.....	25
Running the Server.....	26
To Start the Identity Proxy.....	26
To Run the Server as a Foreground Process.....	26

To Start the Server at Boot Time.....	26
Stopping the Identity Proxy.....	27
To Stop the Server.....	27
To Schedule a Server Shutdown.....	27
To Restart the Server.....	28
Uninstalling the Server.....	28
To Uninstall the Server in Interactive Mode.....	29
To Uninstall the Server in Non-Interactive Mode.....	30
To Uninstall Selected Components in Non-Interactive Mode.....	30
To Uninstall the RPM Build Package.....	30
Updating the Identity Proxy.....	30
To Update the Identity Proxy.....	31
To Upgrade the RPM Package.....	32
Reverting an Update.....	32
Installing the Management Console.....	33
To Install the Management Console Out of the Box.....	33
Working with the Management Console.....	35
Logging into the Management Console.....	35
Fine-Tuning the Management Console.....	37

Chapter 3: Configuring the Identity Proxy..... 41

Overview of the Configuration and Management Tools.....	42
Using the create-initial-proxy-config Tool.....	42
Configuring a Standard Identity Proxy Deployment.....	43
To Configure a Standard Identity Proxy Deployment.....	43
About dsconfig Configuration Tool.....	46
Using dsconfig in Interactive Command-Line Mode.....	46
Using dsconfig Interactive Mode: Viewing Object Menus.....	46
Using dsconfig in Non-Interactive Mode.....	47
Using dsconfig Batch Mode.....	49
Using the Configuration API.....	50
Authentication and Authorization with the Configuration API.....	50
Relationship Between the Configuration API and the dsconfig Tool.....	51
Configuration API Paths.....	51
Updating Properties.....	52
Administrative Actions.....	52
Configuration API Responses.....	52
Generating a Summary of Configuration Components.....	53
To Generate a Summary of Configuration Components.....	53
Configuring Server Groups.....	56
About the Server Group Example.....	56
To Create a Server Group.....	56
Managing Root Users Accounts.....	58
Default Root Privileges.....	58
Configuring Locations.....	61
To Configure Locations Using dsconfig.....	61
To Modify Locations Using dsconfig.....	64
Configuring Batched Transactions.....	65
To Configure Batched Transactions.....	66
Configuring Server Health Checks.....	66
About the Default Health Checks.....	66
About Creating a Custom Health Check.....	67
Configuring LDAP External Servers.....	69
About the prepare-external-server Tool.....	70
To Configure an External Server Using dsconfig.....	72

Configuring Load Balancing.....	73
To Configure Load Balancing Using dsconfig.....	74
Configuring Criteria-Based Load-Balancing Algorithms.....	75
Understanding Failover and Recovery.....	81
Configuring Proxy Transformations.....	82
To Configure Proxy Transformations Using dsconfig.....	83
Configuring Request Processors.....	84
To Configure Request Processors Using dsconfig.....	85
To Pass LDAP Controls with the Proxying Request Processor.....	86
Configuring Server Affinity.....	87
To Configure Server Affinity.....	87
Configuring Subtree Views.....	88
To Configure Subtree View.....	89
Configuring Client Connection Policies.....	89
Understanding the Client Connection Policy.....	90
When a Client Connection Policy is Assigned.....	90
Restricting the Type of Search Filter Used by Clients.....	91
Defining Request Criteria.....	92
Setting Resource Limits.....	92
Defining the Operation Rate.....	92
Client Connection Policy Deployment Example.....	93
Configuring Globally Unique Attributes.....	96
About the Globally Unique Attribute Plug-in.....	96
To Configure the Globally Unique Attribute Plug-in.....	97
Configuring the Global Referential Integrity Plug-in.....	98
Sample Global Referential Integrity Plug-in.....	98
Configuring an Active Directory Server Back-end.....	98

Chapter 4: Managing Access Control..... 101

Overview of Access Control.....	102
Key Access Control Features.....	102
General Format of the Access Control Rules.....	104
Summary of Access Control Keywords.....	104
Working with Targets.....	110
target.....	110
targetattr.....	111
targetfilter.....	113
targettrfilters.....	113
targetscope.....	114
targetcontrol.....	115
extOp.....	115
Examples of Common Access Control Rules.....	115
Administrator Access.....	115
Anonymous and Authenticated Access.....	116
Delegated Access to a Manager.....	116
Proxy Authorization.....	116
Validating ACIs Before Migrating Data.....	117
To Validate ACIs from a File.....	117
To Validate ACIs in Another Identity Proxy.....	118
Migrating ACIs from Sun/Oracle to UnboundID Identity Data Store.....	119
Support for Macro ACIs.....	119
Support for the roleDN Bind Rule.....	119
Targeting Operational Attributes.....	119
Specification of Global ACIs.....	120
Defining ACIs for Non-User Content.....	120

Limiting Access to Controls and Extended Operations.....	120
Tolerance for Malformed ACI Values.....	120
About the Privilege Subsystem.....	121
Identifying Unsupported ACIs.....	121
Working with Privileges.....	121
Available Privileges.....	122
Privileges Automatically Granted to Root Users.....	124
Assigning Additional Privileges for Administrators.....	125
Assigning Privileges to Normal Users and Individual Root Users.....	125
Disabling Privileges.....	125
 Chapter 5: Deploying a Standard Identity Proxy.....	127
Creating a Standard Multi-Location Deployment.....	128
Overview of the Deployment Steps.....	128
Installing the First Identity Proxy.....	129
Configuring the First Identity Proxy.....	130
Defining Locations.....	132
Configuring the External Servers in the East Location.....	133
Apply the Configuration to the Identity Proxy.....	135
Configuring Additional Identity Proxy Instances.....	136
Testing External Server Communications After Initial Setup.....	138
Testing a Simulated External Server Failure.....	139
Expanding the Deployment.....	140
Overview of Deployment Steps.....	141
Preparing Two New External Servers Using the prepare-external-server Tool.....	141
Adding the New UnboundID Identity Data Stores to the Identity Proxy.....	142
Adding New Locations.....	143
Editing the Existing Locations.....	145
Adding New Health Checks for the Central Servers.....	150
Adding New External Servers.....	153
Modifying the Load Balancing Algorithm.....	158
Testing External Server Communication.....	162
Testing a Simulated External Server Failure.....	162
Merging Two Data Sets Using Proxy Transformations.....	163
Overview of the Attribute and DN Mapping.....	163
About Mapping Multiple Source DNs to the Same Target DN.....	164
An Example of a Migrated Sample Customer Entry.....	165
Overview of Deployment Steps.....	165
About the Schema.....	166
Creating Proxy Transformations.....	167
Creating the Attribute Mapping Proxy Transformations.....	167
Creating the DN Mapping Proxy Transformations.....	172
Creating a Request Processor to Manage the Proxy Transformations.....	178
Creating Subtree Views.....	182
Editing the Client Connection Policy.....	183
Testing Proxy Transformations.....	186
 Chapter 6: Deploying an Entry-Balancing Identity Proxy.....	189
Deploying an Entry-Balancing Proxy Configuration.....	190
Determining How to Balance Your Data.....	191
Entry Balancing and ACIs.....	191
Overview of Deployment Steps.....	192
Installing the Identity Proxy.....	192

Configuring the Entry-Balancing Identity Proxy.....	192
Configuring the Placement Algorithm Using a Batch File.....	200
Rebalancing Your Entries.....	201
About Dynamic Rebalancing.....	202
About the move-subtree Tool.....	204
About the subtree-accessibility Tool.....	204
Managing the Global Indexes in Entry-Balancing Configurations.....	204
When to Create a Global Attribute Index.....	205
Reloading the Global Indexes.....	205
Monitoring the Size of the Global Indexes.....	206
Sizing the Global Indexes.....	207
Priming the Global Indexes on Start Up.....	208
Priming or Reloading the Global Indexes from Sun Directory Servers.....	209
Working with Alternate Authorization Identities.....	209
About Alternate Authorization Identities.....	210
Configuring Alternate Authorization Identities.....	211

Chapter 7: Managing Entry-Balancing Replication.....213

Overview of Replication in an Entry-Balancing Environment.....	214
Replication Prerequisites in an Entry-Balancing Deployment.....	214
About the --restricted Argument of the dsreplication Command-Line Tool.....	215
To Use the --restricted Argument of the dsreplication Command-Line Tool.....	215
Checking the Status of Replication in an Entry-Balancing Deployment.....	216
To Check the Status of Replication in an Entry-Balancing Deployment.....	216
Example of Configuring Entry-Balancing Replication.....	216
Assumptions.....	217
Configuration Summary.....	217

Chapter 8: Managing Security..... 223

Summary of the UnboundID Identity Proxy Security Features.....	224
Identity Proxy SSL and StartTLS Support.....	225
LDAP-over-SSL (LDAPS).....	226
StartTLS Support.....	226
Managing Certificates.....	226
Authentication Using Certificates.....	227
Creating Server Certificates using Keytool.....	227
Client Certificates.....	231
Creating PKCS#12 Certificates.....	231
Working with PKCS#11 Tokens.....	232
Configuring the Key and Trust Manager Providers.....	232
Configuring the JKS Key and Trust Manager Provider.....	233
Configuring the PKCS#12 Key Manager Provider.....	234
Configuring the PKCS#11 Key Manager Provider.....	235
Configuring the Blind Trust Manager Provider.....	236
Configuring SSL in the Identity Proxy.....	236
To Configure SSL in the Identity Proxy.....	236
Configuring StartTLS.....	238
To Configure StartTLS.....	238
Authentication Mechanisms.....	239
Simple Authentication.....	239
Working with SASL Authentication.....	239
Working with the SASL ANONYMOUS Mechanism.....	239
Working with the SASL PLAIN Mechanism.....	240

Working with the SASL CRAM-MD5 Mechanism.....	241
Working with the SASL DIGEST-MD5 Mechanism.....	243
Working with the SASL EXTERNAL Mechanism.....	246
Working with the GSSAPI Mechanism.....	247
Working with the UNBOUNDID-TOTP SASL Mechanism.....	251
Working with the UNBOUNDID-DELIVERED-OTP SASL.....	253
Configuring Pass-Through Authentication.....	255
To Configure Pass-Through Authentication.....	256
Adding Operational Attributes that Restrict Authentication.....	257
Configuring Certificate Mappers.....	258
Configuring the Subject Equals DN Certificate Mapper.....	258
Configuring the Fingerprint Certificate Mapper.....	259
Configuring the Subject Attribute to User Attribute Certificate Mapper.....	260
Configuring the Subject DN to User Attribute Certificate Mapper.....	261

Chapter 9: Managing the Identity Proxy..... 263

Managing Logs.....	264
About the Default Logs.....	264
Error Log.....	264
server.out Log.....	266
Debug Log.....	266
Audit log.....	267
Config Audit Log and the Configuration Archive.....	267
Access and Audit Log.....	268
Setup Log.....	269
Tool Log.....	269
LDAP SDK Debug Log.....	269
Types of Log Publishers.....	270
Creating New Log Publishers.....	270
To Create a New Log Publisher.....	271
To Create a Log Publisher Using dsconfig Interactive Command-Line Mode.....	271
About Log Compression.....	272
About Log Signing.....	272
To Configure Log Signing.....	273
To Validate a Signed File.....	273
Configuring Log Rotation.....	273
To Configure the Log Rotation Policy.....	274
Configuring Log Retention.....	274
To Configure the Log Retention Policy.....	274
Setting Resource Limits.....	275
Setting Global Resource Limits.....	275
Setting Client Connection Policy Resource Limits.....	276
Monitoring the Identity Proxy.....	277
Monitoring System Data Using the Metrics Engine.....	277
To Monitor Server Using the Status Tool.....	277
About the Monitor Entries.....	278
Using the Monitoring Interfaces.....	279
Monitoring with the Management Console.....	280
Monitoring with JMX.....	281
Monitoring over LDAP.....	284
Monitoring Using the LDAP SDK.....	284
Monitoring Using SNMP.....	285
Profiling Server Performance Using the Stats Logger.....	289
To Enable the Stats Logger.....	290
To Configure Multiple Periodic Stats Loggers.....	291

Adding Custom Logged Statistics to a Periodic Stats Logger.....	292
Working with Alarms, Alerts, and Gauges.....	294
To Test Alarms and Alerts.....	295
Indeterminate Alarms.....	296
Working with Administrative Alert Handlers.....	297
Configuring the JMX Connection Handler and Alert Handler.....	298
Configuring the SMTP Alert Handler.....	299
Configuring the SNMP Subagent Alert Handler.....	299
Working with Virtual Attributes.....	300
About the Server SDK.....	300
 Chapter 10: Managing Monitoring.....	301
The Monitor Backend.....	302
Monitoring Disk Space Usage.....	303
Monitoring with the Metrics Engine.....	304
Monitoring Key Performance Indicators by Application.....	304
Configuring the External Servers.....	305
Proxy Considerations for Tracked Applications.....	307
Monitoring Using SNMP.....	307
SNMP Implementation.....	307
Configuring SNMP.....	308
Configuring SNMP on AIX.....	311
MIBS.....	311
Monitoring with the Management Console.....	312
To View the Monitor Dashboard.....	312
Accessing the Processing Time Histogram.....	313
Monitoring with JMX.....	313
Running JConsole.....	313
Monitoring the Identity Proxy Using JConsole.....	315
Monitoring Using the LDAP SDK.....	317
Monitoring over LDAP.....	317
Profiling Server Performance Using the Stats Logger.....	318
To Enable the Stats Logger.....	318
To Configure Multiple Periodic Stats Loggers.....	319
Adding Custom Logged Statistics to a Periodic Stats Logger.....	320
 Chapter 11: Troubleshooting the Identity Proxy.....	323
Garbage Collection Diagnostic Information.....	324
Working with the Troubleshooting Tools.....	324
Working with the Collect Support Data Tool.....	324
Identity Proxy Troubleshooting Tools.....	326
Server Version Information.....	326
LDIF Connection Handler.....	326
Embedded Profiler.....	327
Troubleshooting Resources for Java Applications.....	327
Java Troubleshooting Tools.....	328
Java Diagnostic Information.....	330
Java Troubleshooting Tools (IBM JDK).....	331
Troubleshooting Resources in the Operating System.....	331
Identifying Problems with the Underlying System.....	332
Monitoring System Data Using the Metrics Engine.....	332
Examining CPU Utilization.....	332
Examining Disk Utilization.....	333

Examining Process Details.....	334
Tracing Process Execution.....	335
Examining Network Communication.....	336
Common Problems and Potential Solutions.....	337
General Methodology to Troubleshoot a Problem.....	337
The Server Will Not Run Setup.....	338
The Server Will Not Start.....	339
The Server Has Crashed or Shut Itself Down.....	342
The Server Will Not Accept Client Connections.....	343
The Server is Unresponsive.....	344
The Server is Slow to Respond to Client Requests.....	345
The Server Returns Error Responses to Client Requests.....	346
The Server Must Disconnect a Client Connection.....	347
Problems with the Management Console.....	348
Problems with the Management Console: JVM Memory Issues.....	348
Global Index Growing Too Large.....	349
Forgotten Proxy User Password.....	349
Providing Information for Support Cases.....	349
 Chapter 12: Managing the SCIM Servlet Extension.....	351
Overview of SCIM Fundamentals.....	352
Summary of SCIM Protocol Support.....	352
About the Identity Access API.....	353
Creating Your Own SCIM Application.....	353
Configuring SCIM.....	354
Before You Begin.....	354
Configuring the SCIM Servlet Extension.....	354
Configuring LDAP Control Support on All Request Processors (Proxy Only).....	356
SCIM Servlet Extension Authentication.....	356
Verifying the SCIM Servlet Extension Configuration.....	358
Configuring Advanced SCIM Extension Features.....	358
Managing the SCIM Schema.....	358
Mapping SCIM Resource IDs.....	365
Using Pre-defined Transformations.....	365
Mapping LDAP Entries to SCIM Using the SCIM-LDAP API.....	366
SCIM Authentication.....	366
SCIM Logging.....	367
SCIM Monitoring.....	367
Configuring the Identity Access API.....	367
To Configure the Identity Access API.....	368
To Disable Core SCIM Resources.....	368
To Verify the Identity Access API Configuration.....	368
Monitoring the SCIM Servlet Extension.....	369
Testing SCIM Query Performance.....	369
Monitoring Resources Using the SCIM Extension.....	370
About the HTTP Log Publishers.....	371
 Chapter 13: Managing Server SDK Extensions.....	373
About the Server SDK.....	374
Available Types of Extensions.....	374
 Chapter 14: Command-Line Tools.....	377

Using the Help Option.....	378
Available Command-Line Utilities.....	378
Managing the tools.properties File.....	380
Creating a Tools Properties File.....	381
Tool-Specific Properties.....	381
Specifying Default Properties Files.....	382
Evaluation Order Summary.....	382
Evaluation Order Example.....	382
Running Task-based Utilities.....	383

Preface

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

Purpose of This Guide

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

The Identity Proxy is part of the UnboundID Platform. The UnboundID Platform is the consumer-grade identity access and management platform—built specifically to handle the massive scale and real-time demands of hundreds of millions of customers. It delivers a consistent, seamless, personalized brand experience that makes each customer feel valued.

The UnboundID Platform provides a unified view of customer data across all applications, channels, partners, and lines of business. The result is:

- Increased customer trust and confidence through greater transparency and customer control of personal data.
- A consistent, personalized customer experience that promotes better conversion, up-selling, and cross-selling.

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:

- Identity Platforms 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® Identity Data Store Administration Guide*
- *UnboundID® Identity Data Store Reference Guide (HTML)*
- *UnboundID® Identity Proxy Administration Guide*
- *UnboundID® Identity Proxy Reference Guide (HTML)*

- *UnboundID® Identity Data Sync Administration Guide*
- *UnboundID® Identity Data Sync Reference Guide (HTML)*
- *UnboundID® Metrics Engine Administration Guide*
- *UnboundID® Identity Broker Administration Guide*
- *UnboundID Security Guide*
- *UnboundID® LDAP SDK*
- *UnboundID® Server SDK*

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
<i>Italic</i>	Identifies file names, doc titles, terms, variable names, and emphasized text

Chapter

1 Introduction

UnboundID® Identity Proxy is a fast and scalable LDAPv3 gateway for the UnboundID® Identity Data Store. The Identity Proxy architecture can be configured to control how client requests are routed to backend servers.

This chapter provides an overview of the Identity Proxy features and components. It contains the following sections:

Topics:

- [*Overview of the UnboundID Identity Proxy Features*](#)
- [*Overview of the Identity Proxy Components and Terminology*](#)
- [*Server Component Architecture*](#)
- [*Identity Proxy Configuration Overview*](#)

Overview of the UnboundID Identity Proxy Features

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

The UnboundID Identity Proxy provides the following set of features:

- **High availability.** The Identity Proxy 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 Identity Proxy can follow them on the client's behalf.
- **Data mapping and transformation.** The Identity Proxy can do DN mapping and attribute mapping to allow clients to interact with the server using older names for directory content. It allows clients to continue working when they would not be able to work directly with the Identity Data Store, either because of changes that have occurred at the data layer or to inherent design limitations in the clients.
- **Horizontal scalability and performance.** Reads can be horizontally scaled using load balancing. In large data centers, if the data set is too large to be cached or to provide horizontal scalability for writes, the Identity Proxy can automatically split the data across multiple systems. This feature allows the Identity Proxy to improve scalability and performance of the Identity Data Store environment.
- **Load balancing and failover.** You can spread the workload across multiple identity proxies in a large data center using load-balancing algorithms. Load balancing is also useful when a server becomes degraded or non-responsive, because client process requesting is directed to a different server.
- **Security and access control.** The Identity Proxy can add additional firewall capabilities, as well as constraints and filtering to help protect the Identity Data Store from attacks. You can use an Identity Proxy in a DMZ as opposed to allowing clients to directly access the Identity Proxy 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 Identity Proxy contains controls that allow administrators to track requests back to the client that issued them. Whenever the Identity Proxy forwards a request to the Identity Data Store, it includes a control in the request so that the Identity Data Store's access log has the IP address of the client, address and connection ID of the Identity Data Store. In the response back to the client, it similarly includes information about the Identity Data Store 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 Identity Proxy uses many of the components of the UnboundID Identity Data Store, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, schema, and so on. You can use the Metrics Engine, the `dsconfig` tool and the Web-based administration console to manage the Identity Proxy.
- **Multi-Platform Support.** The UnboundID Identity Proxy 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 Identity Proxy Components and Terminology

The Identity 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 Identity Proxy 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 Identity Data Store 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 Identity Proxy instances should have the same configuration except for the attribute that specifies the location of the Identity Proxy 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 identity data store instances accessed by the UnboundID Identity Proxy. 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 Identity Proxy can recognize this and accommodate it.

The UnboundID Identity Proxy 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 Identity Proxy can be configured to prefer the server with the higher score.

The Identity Proxy 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 Identity Proxy 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 Identity Proxy are made available to the load-balancing algorithms so that they may be taken into account when determining where to send requests. The Identity Proxy 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 your 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 Identity Proxy 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 Identity Proxy does not flip back and forth between available and degraded.

UnboundID Identity Proxy 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 Identity Proxy 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 Identity Data Store administrative alerts.** If the Identity Data Store indicates there is a problem, for example an index that must be rebuilt, then it will flag itself as degraded or unavailable. When the Identity Proxy detects this, it will stop sending requests to the server. The Identity Proxy detects administrative alerts as soon as they are issued by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Identity Data Store. When the Identity Proxy is notified by the Identity Data Store of a new alert, it immediately retrieves the base `cn=monitor` entry of the Identity Data Store. If this

entry has a value for the `unavailable-alert-type` attribute, then the Identity Proxy will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Identity Proxy will consider it degraded. Clients of the Identity Proxy can use a similar mechanism to detect and react when an Identity Proxy flags itself as degraded or unavailable.

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

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

About Load-Balancing Algorithms

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

- **Consider the location of the server.** Servers in the same location as the Identity Proxy 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 identity data store 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 Identity Proxy 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 Identity Proxy. 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 identity data store handling the request uses `dc=example,dc=com`. The Identity Proxy 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 Identity Proxy can map it to `uid` before sending the request on to the backend LDAP server. The Identity Proxy then remaps the response to `userID` when the value is returned.

The Identity Proxy 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 Identity Proxy provides the following types of request processor:

- **Proxying request processors**, which forward operations received by the Identity Proxy 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.

Identity Proxy 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 Identity Proxy 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 Identity Proxy establishes connections, it authenticates them using whatever authentication mechanism is defined in the configuration of the external server. These connections will be re-used for all types of operations to be forwarded to the backend server. The bind DN and password are configured in the Identity Proxy.

Whenever a client sends a bind request to the Identity Proxy, 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 Identity Proxy itself and it will not be forwarded to the backend server. However, the Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Data Store 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 identity data stores for improved scalability and performance. Entry balancing

can take advantage of a global index, an in-memory cache used to quickly determine which set or sets of servers should be used to process a request based on the entry DNs and/or the attribute values used in the request.

For information about configuring entry balancing, see [Deploying an Entry-Balancing Proxy Configuration](#) on page 190.

Server Component Architecture

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

Architecture of a Simple Identity Proxy Deployment

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

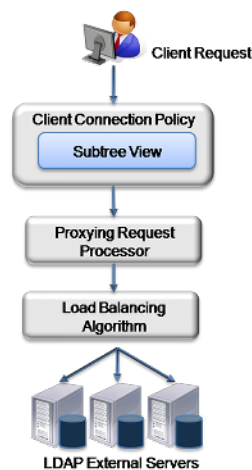


Figure 1: Process Flow for Identity Proxy

The client connection policy contains a subtree view, which defines the portion of the DIT available to clients. Once the Identity Proxy 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 Identity Proxy Deployment

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

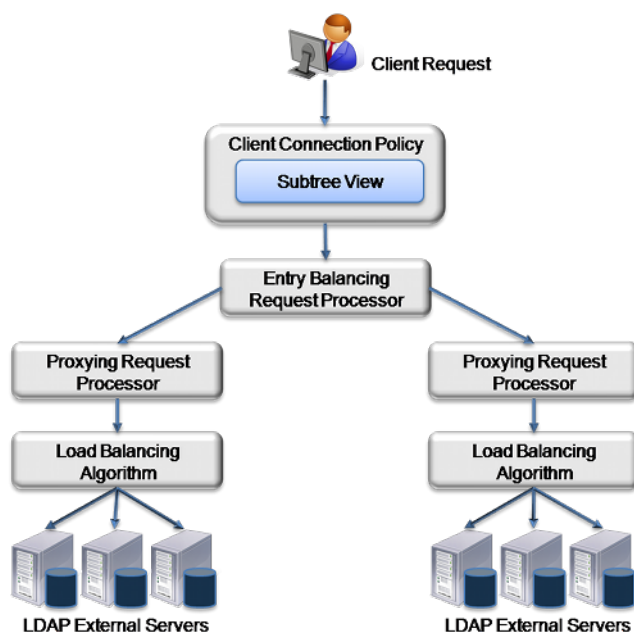


Figure 2: Process Flow for Entry-Balancing Identity Proxy

Entry balancing is typically used when the data set is too large to fully cache on a single server or when the write performance requirements of an environment are higher than can be achieved with a single replicated set of servers. In such cases, the data 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 Identity Proxy communicates with a set of clients. It contains a subtree view that represents the base DN for the entire deployment. The data set splits beneath this base DN.

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

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

For configuration information, see [Configuring an Entry-Balancing Identity Proxy Deployment](#). For information about entry-balancing replication, see [Overview of Replication in an Entry-Balancing Environment](#) on page 214.

Identity Proxy Configuration Overview

The configuration of the Identity Proxy 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 Identity Proxy. 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 Identity Proxy location.** You need to update the configuration to specify the location of the Identity Proxy instance.
- **Configuring health checks for the LDAP external servers.** You can configure at what point the Identity Proxy 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 Identity Data Stores, Sun Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured in the previous step.
- **Configuring the load-balancing algorithm.** You configure the load-balancing algorithm used by the Identity Proxy to determine which server in a set of similar servers should be used to process a client request. The Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 Identity Proxy

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

It includes the following sections:

Topics:

- [*Before You Begin*](#)
- [*Preparing the Operating System*](#)
- [*Getting the Installation Packages*](#)
- [*Installing the Identity Proxy*](#)
- [*Running the Server*](#)
- [*Stopping the Identity Proxy*](#)
- [*Uninstalling the Server*](#)
- [*Updating the Identity Proxy*](#)
- [*Installing the Management Console*](#)
- [*Working with the Management Console*](#)

Before You Begin

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

Important:



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

Defining a Naming Strategy for Server Locations

The various objects defined in the UnboundID Identity Proxy will be specific to a particular location. Location names are used to define a grouping of UnboundID Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.

Software Requirements: Java

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

Even if your system already has Java installed, you may want to create a separate Java installation for use by the UnboundID Identity Proxy to ensure that updates to the system-wide Java installation do not inadvertently impact the Identity Proxy. This setup requires that the JDK, rather than the JRE, for the 64-bit version, 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 64-bit version of Java contains a complete installation. 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 Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

To Install Java (IBM)

1. Open a browser and navigate to the IBM download site.
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 Identity Proxy will run.

Configuring the File Descriptor Limits

The UnboundID Identity Proxy allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. The number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTOR` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTOR=12345`. If these are not set, the default of 65535 is used.

If an error occurs while trying to use the configured number of file descriptors, it is written to the terminal. If the error occurs at startup, a similar message is written to the server's error log.

To Set the Filesystem Flushes

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

```
commit=1
```

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

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

About Editing OS-Level Environment Variables

Certain environment variables can impact the Identity Proxy 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 Identity Proxy 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.

Install `dstat` (SUSE Linux)

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

1. Login as Root.
2. Add the appropriate repository using the `zypper` tool.

```
$ zypper addrepo http://download.opensuse.org/repositories/server:/monitoring/SLE_11_SP2 Monitoring
```

3. Install the `dstat` utility.

```
$ zypper install dstat
```

Omit `vm.overcommit_memory`

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

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

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

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 Identity Proxy code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Identity Proxy 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 Identity Proxy.

About the RPM Package

UnboundID supports the UnboundID Identity Proxy release bundle in an RPM Package Manager (RPM) package for customers who require it. By default, the RPM unpacks the code at `/opt/unboundid/proxy`, after which you can run the `setup` command to install the server at that location.

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

To Install the RPM Package

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

```
$ rpm --install unboundid-proxy-<version>.rpm
```

3. From `/opt/unboundid/proxy/UnboundID-Proxy`, run the `setup` command to install the server on the machine.

Installing the Identity Proxy

When you deploy UnboundID Identity Proxy 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 Identity Proxy allows you to clone a proxy configuration. First, you install an Identity Proxy 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 identity proxies in your topology.

About the setup Tool

One of the strengths of the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy will run.

Installing the First Identity Proxy in Interactive Mode

The `setup` tool provides an interactive text-based interface to install an Identity Proxy instance.

To Install the First Identity Proxy in Interactive Mode

1. Change to the server root directory.

```
$ cd UnboundID-Proxy
```

2. Use the `setup` command to install the Identity Proxy 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 fully-qualified host name or IP address of the host machine.
6. Enter the root user DN, or press **Enter** to accept the default (`cn=Directory Manager`), and then type and confirm the root user password.
7. Choose the communication option for SCIM and the Configuration API. HTTPS is recommended.
8. Enter the port to accept connections from HTTPS clients. For this example, press **Enter** to accept the default (8443). Note that the HTTPS/SCIM URL will be `https://<hostname>:8443/`.
9. Enter the LDAP port number of your Identity Proxy, or press **Enter** to accept the default port, which is 389.

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

10. For enabling LDAPS, enter "yes", and then enter the port to accept connections from the Identity Proxy. For this example, press **Enter** to accept the default LDAPS port.

```
Do you want to enable SSL? (yes / no) [no]: yes
On which port would you like the Identity Proxy to accept connections from LDAPS
clients? [3636]:
```

11. For StartTLS, press **Enter** to accept the default (no).

- 12.**For certificate options, select the certificate option for the server. For this example, press **Enter** to accept the default (generate self-signed certificate). For actual deployments, you will likely use an existing certificate.

```
Certificate server options:
```

- ```
1) Generate self-signed certificate (recommended for testing purposes only)
2) Use an existing certificate located on a Java KeyStore (JKS)
3) Use an existing certificate located on a PKCS#12 keystore
4) Use an existing certificate on a PKCS#11 token
```

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

- 13.**By default, the Identity Proxy 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):
```

- 14.**To configure your Identity Proxy to use entry balancing, where leaf and non-leaf entries below a common parent entry are distributed among multiple identity data stores, 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 Identity Proxy. 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.

---

- 15.**If you want to configure an entry balanced Identity Proxy topology, enter `yes`. Otherwise, accept the default of `no`.

- 16.**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 Identity Proxy 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]:
```

- 17.**If you choose to tune the JVM, enter the maximum amount of memory you want the Identity Proxy to allocate to the server and tool. In this example, the maximum allowed for the server is 16 gigabytes.

The command line provides a dynamic value range based on the resources of the system on which the installer is running.

```
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]:
```

18. To start the server after the configuration has completed, type `yes`, or press **Enter** to accept the default. 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]:
```

19. 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
=====
Root User DN: cn=Directory Manager
LDAP Listener Port: 389
HTTP Listener Port: disabled
Secure Access: Enable SSL on LDAP Port 3636
 Enable SSL on HTTP Port 8443
 Create a new self-signed certificate
 Generate default trust store
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]:
```

20. Once setup is complete, you are prompted to begin the server 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 Identity Proxy later. You need to configure your Identity Proxy later if you plan to use custom schema in your deployment. For more information about configuring your Identity Proxy 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 Identity Proxy Instances in Interactive Mode

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

1. Change to the server root directory.

```
$ cd UnboundID-Proxy
```

2. Use the `setup` command to install the Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy.

```
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 Identity Proxy, 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 Identity Proxy.

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

10. Choose the location of your new Identity Proxy 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 fully-qualified host name or IP address of the host machine.

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

**13.** Choose the communication option for SCIM and the Configuration API. HTTPS is recommended.

**14.** Enter the port to accept connections from HTTPS clients. For this example, press Enter to accept the default (8443). Note that the HTTPS/SCIM URL will be `https://<hostname>:8443/`.

**15.** Enter the LDAP port number of your Identity Proxy, or press **Enter** to accept the default port, which is 389.

```

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

```

**16.** For enabling LDAPS, enter "yes", and then enter the port to accept connections from the Identity Proxy. For this example, press Enter to accept the default LDAPS port.

```

Do you want to enable SSL? (yes / no) [no]: yes
On which port would you like the Identity Proxy to accept connections from LDAPS
clients? [3636]:

```

**17.** For StartTLS, press Enter to accept the default (no).

**18.** For certificate options, select the certificate option for the server. For this example, press Enter to accept the default (generate self-signed certificate). For actual deployments, you will likely use an existing certificate.

```

Certificate server options:

1) Generate self-signed certificate (recommended for testing purposes only)
2) Use an existing certificate located on a Java KeyStore (JKS)
3) Use an existing certificate located on a PKCS#12 keystore
4) Use an existing certificate on a PKCS#11 token

Enter choice [1]: 2
Java KeyStore (JKS) path: /path/to/keystore
KeyStore PIN:

```

**19.** By default, the Identity Proxy 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 Identity Proxy 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 Identity Proxy and tools as follows:

```
--aggressiveJVMtuning --maxHeapSize 256m
```

If you are using entry balancing, tune the amount of memory devoted to the Identity Proxy 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 KeyStore is expected to be the same as the password to the KeyStore. If this is not the case, the private key password can be defined within the 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 Identity Proxy in Non-Interactive Mode

- Use `setup` with the `--no-prompt` option. The command uses the default root user DN (`cn=Directory 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy with a Truststore in Non-Interactive Mode

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



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

## To Install the Identity Proxy with a Truststore in Non-Interactive Mode

- Run the `setup` tool to install an Identity Proxy 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 ..... Done

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 Identity Proxy, 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 Identity Proxy as a background process when no options are specified. To run the Identity Proxy as a foreground process, use the `bin/start-proxy` command with the `--nodetach` option.

### To Start the Identity Proxy

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 Identity Proxy as a foreground process.

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

2. You can stop the Identity Proxy 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 Identity Proxy does not start automatically when the system is booted. Instead, you must manually start it with the `bin/start-proxy` command. To configure the Identity Proxy 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 (starting with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

```
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 Identity Proxy

The Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 server root 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
```

## Uninstalling the Server

The Identity Proxy 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 Identity Proxy instance, you can also manually remove the Identity Proxy 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 Identity Proxy 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 Identity Proxy.

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

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

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

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

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

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

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

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

4. If the Identity Proxy 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 Identity Proxy
on server.example.com:389?

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

Enter choice [3]:
```

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

## To Uninstall the Server in Non-Interactive Mode

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

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

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

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

## To Uninstall Selected Components in Non-Interactive Mode

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

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

## To Uninstall the RPM Build Package

1. From the server root directory, remove the RPM package use the `--erase` option with the `<rpm-id>`. The `<rpm-id>` is `unboundid-proxy` and removes the files at `/opt/unboundid/proxy/UnboundID-Proxy`.

```
$ rpm --erase unboundid-proxy
```

2. The `rpm` command specifies if any files or directories require manual deletion. Manually remove any remaining directories or files using `rm -rf <directory>`.

# Updating the Identity Proxy

UnboundID issues new software builds periodically and distributes the software package in zip format. Administrators can use the Identity Proxy's `update` utility to update the current server code with the latest features and bug fixes. To update the Identity Proxy 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 Identity Proxy.

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 Identity Proxy

Assume that an existing version of the Identity Proxy 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 Identity Proxy build. Also, make sure you have a clear backout plan and schedule.
2. Download the latest version of the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy 5.0.1.0
Build 2011043200609Z Revision 9235
```

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

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



**Note:** The UnboundID Identity Proxy provides a web console called the Management Console, to configure and monitor the server. If you update the Identity Proxy version, you should also update the 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 Identity Proxy version number and revision.

```
$ view <server-root>/history/1272307020420-5.0.1.0.9235/update.log
```

### To Upgrade the RPM Package

If the Linux RPM package was used to install the Identity Data Store, the following should be performed to upgrade the server.

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

```
$ rpm --upgrade unboundid-proxy-<new-version>.rpm
```

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

The upgrade history is written to `/opt/unboundid/proxy/UnboundID-Proxy/history/<timestamp>/update.log`.

### Reverting an Update

Once the Identity Proxy 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 Identity Proxy, 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 Identity Proxy 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
```

### Configure SCIM After Upgrade

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

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

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

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

```
dsconfig set-request-processor-prop \
 --processor-name dc_example_dc_com-req-processor \
 --add supported-control-oid:1.2.840.113556.1.4.1413
```

In the last command, `dc_example_dc_com-req-processor` is the default processor name. Replace it with the correct name for your system.

## Installing the Management Console

The UnboundID Identity Proxy provides a graphical web application tool, the UnboundID Management Console. The 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 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 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 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 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.
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 Management Console ZIP file, `UnboundID-Proxy-web-console-5.0.1.0.zip` and unzip the file on your local host. You should see the following files:

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

4. Create a `proxyconsole` directory in `apache-tomcat-<version>/webapps/proxyconsole`. Then, copy the `proxyconsole.war` file to `apache-tomcat-<version>/webapps/proxyconsole`. If the servlet is running and auto-deploy is enabled, copy the `.war` file to the `/webapps` directory and it will install in the directory.

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

6. Optional. Edit the `WEB-INF/web.xml` file to point to the correct Identity Proxy 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>
```

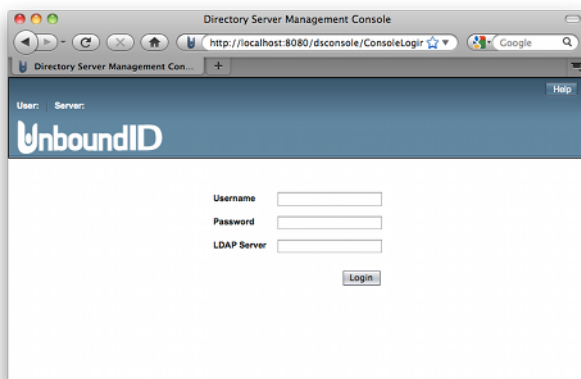
The session expires after the specified number of minutes. Changing the value to 120, for example, will extend the expiration to two hours. Changes to this setting might not take effect until the servlet container is restarted, so consider changing the value before starting the server for the first time.

8. Start the Identity Proxy if it is not already running, and then start the 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 Identity Proxy, you must also log out of the current Management Console session and then log back in to start a new console session.

## Working with the Management Console

The Management Console does not persistently store any credentials for authenticating to the Identity Proxy 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 Identity Proxy.

### Logging into the 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 Management Console

1. Go to the server root directory.

```
$ cd UnboundID-Proxy
```

2. Start the Identity Proxy.

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

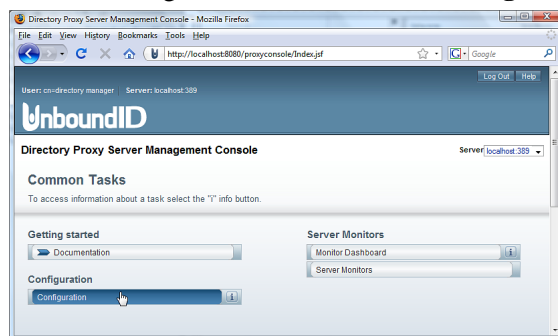
3. Start the Apache Tomcat application server.

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

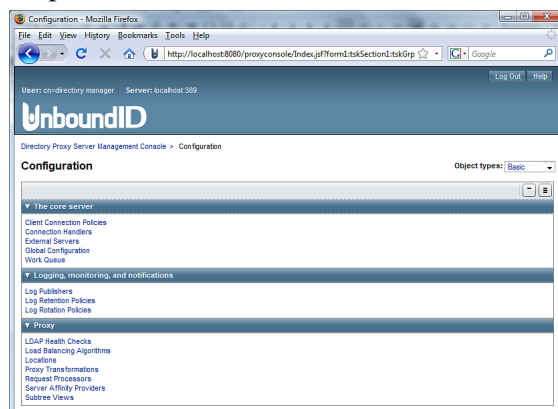
4. Open a browser to `http://hostname:8080/proxyconsole/`.

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

6. On the 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 Management Console

1. Shut down the console and servlet container.
2. In the current deployment of the Management Console, move the `webapps/proxyconsole/WEB-INF/web.xml` file to another location.
3. Download and deploy the latest version for the 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 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 Management Console's `web.xml` to `webapps/proxyconsole/WEB-INF/web.xml`.
5. Start the servlet container.

## To Uninstall the Management Console

1. Close the 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.

## Fine-Tuning the Management Console

The Management Console uses a `web.xml` descriptor file for its configuration and deployment settings. Instead of specifying the host name and port on the Login page, you can configure one or more primary servers in the `web.xml` file as well as configure security and truststore settings for your Identity Proxy console. If you specify any servers using the `web.xml` file, the Login page will no longer display the LDAP Server field. It will automatically attempt to connect to the primary server(s) specified in the `web.xml` file in the order in which they are specified until one of the servers can authenticate the username and password. The console also uses this server to "discover" other servers in the topology, making them available for monitoring and management in the console.

## To Configure One or More Primary Servers for the Console

1. Open the `proxyconsole/WEB-INF/web.xml` file in a text editor to specify the server(s) that the console uses to authenticate. First, remove the comment tags (`<!--` and `-->`) in the `ldap-servers` section.
2. Next, specify the servers as `host:port` (e.g., `server1.example.com:389`) or using the LDAPS protocol to specify security information (e.g., `ldaps://server1.example.com:389`). If you specify more than one server, you must separate them using a space. For example, if you have two servers: one using standard LDAP communication, the other using SSL, you would see the following:

```
<context-param>
 <param-name>ldap-servers</param-name>
 <param-value>localhost:389 ldaps://svr1.example.com:389</param-value>
</context-param>
```

3. Save the file.

## To Configure SSL for the Primary Console Server

You can configure the console so that it will communicate with all of its primary servers over SSL or StartTLS. See the previous section on how to specify one or more primary servers.

1. Open the `proxyconsole/WEB-INF/web.xml` file in a text editor to specify the type of communication to authenticate. First, remove the comment tags (`<!--` and `-->`) in the security section.
2. Specify `none`, `ssl`, or `starttls` for the type of security that you are using to communicate with the Identity Proxy.

```
<context-param>
 <param-name>security</param-name>
 <param-value>ssl</param-value>
</context-param>
```

3. Save the file.

## To Configure a Generic Error Page

The console can be configured so that it displays a generic error page that does not contain sensitive LDAP or server information. If configured, server logs will still contain detailed error information.

1. Open the `proxyconsole/WEB-INF/web.xml` file in a text editor to specify.
2. To display a generic error page in the Management Console, change the value of the following configuration parameter to `false`:

```
<context-param>
 <param-name>detailedErrorMessages</param-name>
 <param-value>>false</param-value>
```

```
</context-param>
```

3. Save the file.

### To Configure a Truststore for the Console

For SSL and StartTLS communication, you can specify your truststore and its password (or password file) in the `web.xml` file. If no truststore is specified, all server certificates will be blindly trusted.

1. Open the `proxyconsole/WEB-INF/web.xml` file in a text editor to specify the truststore. First, remove the comment tags (`<!--` and `-->`) in the truststore section.
2. Specify the path to your truststore.

```
<context-param>
 <param-name>trustStore</param-name>
 <param-value>/path/to/truststore</param-value>
</context-param>
```

3. Next, specify the password or the path to the password pin file.

```
<context-param>
 <param-name>trustStorePassword</param-name>
 <param-value>password</param-value>
</context-param>

<context-param>
 <param-name>trustStorePasswordFile</param-name>
 <param-value>/path/to/truststore/pin/file</param-value>
</context-param>
```

4. Save the file.



## Chapter

# 3

## Configuring the Identity Proxy

---

Once you have initially configured the UnboundID Identity Proxy, 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\*](#)
- [\*Using the create-initial-proxy-config Tool\*](#)
- [\*Configuring a Standard Identity Proxy Deployment\*](#)
- [\*About dsconfig Configuration Tool\*](#)
- [\*Using the Configuration API\*](#)
- [\*Generating a Summary of Configuration Components\*](#)
- [\*Configuring Server Groups\*](#)
- [\*Managing Root Users Accounts\*](#)
- [\*Configuring Locations\*](#)
- [\*Configuring Batched Transactions\*](#)
- [\*Configuring Server Health Checks\*](#)
- [\*Configuring LDAP External Servers\*](#)
- [\*Configuring Load Balancing\*](#)
- [\*Configuring Proxy Transformations\*](#)
- [\*Configuring Request Processors\*](#)
- [\*Configuring Server Affinity\*](#)
- [\*Configuring Subtree Views\*](#)
- [\*Configuring Client Connection Policies\*](#)
- [\*Configuring Globally Unique Attributes\*](#)
- [\*Configuring the Global Referential Integrity Plug-in\*](#)
- [\*Configuring an Active Directory Server Back-end\*](#)

## Overview of the Configuration and Management Tools

The UnboundID Identity Proxy 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 Identity Proxy. We strongly recommend that you use the `create-initial-proxy-config` tool for your initial Identity Proxy 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 Identity Proxy.
- **Using the Management Console.** The UnboundID Identity Proxy 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 Identity Proxy and the external directory server. This tool can be used in conjunction with `create-initial-proxy-config` or `dsconfig` to simplify configuring your Identity Proxy 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 Identity Proxy. You are prompted to launch this tool after installing the Identity Proxy. 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 Identity Data Store, 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 identity data stores 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 Identity Proxy's `logs` directory.

You can only run the `create-initial-proxy-config` tool once for the initial configuration of each Identity Proxy instance. To tune your configuration, use the `dsconfig` tool. When installing a second Identity Proxy, it will not be necessary to run the `create-initial-proxy-config` tool again, as the Identity Proxy 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 Identity Proxy deployment as well as an entry balancing configuration.

## Configuring a Standard Identity Proxy Deployment

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

### To Configure a Standard Identity Proxy Deployment

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

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

2. The initial proxy configuration presents the assumptions about the underlying Identity Data Store backend servers. If the servers do not meet the requirements, then you can enter "no" to quit the process.

```
Some assumptions are made about the topology in order to keep this tool simple:
```

- ```
1) all servers will be accessible via a single user account
2) all servers support the same communication security type
3) all servers are UnboundID, Alcatel-Lucent 8661, Sun
   Java System 5.x, 6.x, or 7.x, or Red Hat (including Fedora and 389)
   directory servers
```

```
If your topology does not have these characteristics you can use this tool to define
a basic configuration and then use the 'dsconfig' tool or the web console to
fine tune the configuration.
```

```
Continue? (yes / no) [yes]:
```

3. Enter the DN for the Identity Proxy user account, then enter and confirm the password for this account. Note that you should not use `cn=Directory Manager` account for communication between the Identity Proxy and the Identity Data Store. For security reasons, the account used to communicate between the Identity Proxy and the Identity Data

Store should not be directly accessible by clients accessing the Identity Proxy. 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 Identity Data Store instances.

```
>>>> External Server Communication Security

Specify the type of security that the Identity Proxy will use when communicating with
directory server instances:

1)  None
2)  SSL
3)  StartTLS

b)  back
q)  quit

Enter choice [1]:
```

5. Specify the base DN of the Identity Data Store instances that will be accessed through the Identity Proxy. The Identity Proxy will create subtree views using each base DN to define portions of the external servers' DIT available for client access. You can specify more than one base DN. Press **Enter** when you have finished specifying the DN(s).

```
Enter a base DN of the directory server instances that will be accessed through the
Identity Proxy:

b)  back
q)  quit

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

6. Next, specify if the entries under your defined subtree view will be split across multiple servers in an entry balanced deployment. For this example, press Enter to accept the default ("no").
7. Define a location for your server, such as the name of your data center or the city where the server is located. This example illustrates defining a location named east.

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

8. If you defined more than one location, specify the location that contains the Identity Proxy itself.

```
Choose the location for this Directory Proxy Server

1)  east
2)  west

b)  back
q)  quit

Enter choice [1]: 1
```

9. Define the hostname:port used by the LDAP external servers. If you have specified more than one location, you will go through this process for each location.

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

10. After each step, the server will attempt to prepare each external server by testing the communication between the Identity Proxy and the Identity Data Store. Select the option "Yes, and all subsequent servers" to indicate that you want the tool to create a proxy user account on all of your LDAP external servers within that location.

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

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

Enter choice [1]: 3
```

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

```
Would you like to create or modify root user 'cn=Proxy User' so that it is available
for this Proxy Server? (yes / no) [yes]:
Enter the DN of an account on ldap-east-01.example.com:389 with which to create or
manage the 'cn=Proxy
User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

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

12. Repeat steps 9-12 for the servers in the other location. Then, press **Enter** to finish configuring the location.

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

```
>>>> Configuration Summary

External Server Security:  SSL
Proxy User DN:            cn=Proxy User,cn=Root DNs,cn=config

Location east
  Failover Order: west
  Servers: localhost:1636

Location west
  Failover Order: east
  Servers: localhost:2636

Base DN: dc=example,dc=com
Servers: localhost:1636, localhost:2636

b) back
q) quit
w) write configuration file

Enter choice [w]:
```

14. Next, apply the configuration changes locally to the Identity Proxy. If you have any Server SDK extensions, make sure to run the `manage-extension` tool, then press **Enter** to apply the changes to the Identity Proxy. 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 Identity Proxy again. Instead, use the `dsconfig` tool.

```
This tool can apply the configuration changes to the local Identity Proxy. This
requires any configured Server SDK extensions to be in place. Do you want to do
this? (yes / no) [yes]:
```

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

About dsconfig Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Identity Data Store 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 Identity Proxy. 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 Identity Proxy, 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 Identity Proxy 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 Identity Proxy in Interactive Mode.
2. On the **UnboundID Identity Proxy 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 Identity Proxy components.

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

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

You can view the `logs/config-audit.log` file to review the configuration changes made to the Identity Proxy 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 Identity Proxy also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to UnboundID Identity Proxy machines.

To Configure the Identity Proxy 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 Identity Proxy. 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  
  
""""
```

Using the Configuration API

UnboundID servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API features a REST-ful design and uses JSON as a text exchange format, so all request headers should allow the application/json content type.

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

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

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, use a secure connection handler such as the HTTPS Connection Handler, for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

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

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

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

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

Relationship Between the Configuration API and the `dsconfig` Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names and formats for configuration object types. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`. Operations supported by the API are those typically found in REST APIs:

The `OPTIONS` method can also be used to determine the operations permitted for a particular path. Use of the `PUT` method for object creation or modification is not supported.

| HTTP Method | Description | Related <code>dsconfig</code> Example |
|-------------|---|---|
| GET | Lists the properties of an object when used with a path representing an object, such as <code>/config/global-configuration</code> or <code>/config/backends/userRoot</code> . Can also list instances objects when used with a path representing a parent relation, such as <code>/config/backends</code> . | <code>get-backend-prop</code> , <code>list-backends</code> , <code>get-global-configuration-prop</code> |
| POST | Creates a new instance of an object when used with a relation parent path, such as <code>/config/backends</code> . | <code>create-backend</code> |
| PATCH | Updates the properties of an existing object when used with a path representing an object, such as <code>/config/backends/userRoot</code> . | <code>set-backend-prop</code> , <code>set-global-configuration-prop</code> |
| DELETE | Deletes an existing object when used with a path representing an object, such as <code>/config/backends/userRoot</code> . | <code>delete-backend</code> |

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

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

Configuration API Paths

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

The schema's paths element enumerates all available sub-paths. The path `/config/backends` in the example can be used to get a listing of existing backends as well as create new ones. A path containing an object name like `/config/backends/{backendName}`, where `{backendName}`

corresponds to an existing backend (such as `userRoot`) may be used to obtain an object's properties, update the properties, or delete the object.

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

Updating Properties

Configuration object properties can be modified with an HTTP PATCH request on a path representing an object or singleton object. Request bodies must consist of a JSON object enumerating a property operation along with a set of property/value pairs, or a list of properties. Operations correspond with and behave the same as the options used in `dsconfig set-{object}-prop` subcommands. Multiple properties can be specified for each operation.

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and the request body will contain an encoded set of actions to be performed.

Configuration API Responses

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

| Response Code | Description | Response Body |
|------------------|---|--|
| 200 Success | The requested operation succeeded, with the response body containing the requested data, or further actions that are required. | List of objects, or object properties, administrative actions. |
| 201 Created | The requested operation to create a new object succeed. A link to the newly created object is sent in the Location response header field. | None. |
| 204 No Content | The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation. | None. |
| 400 Bad Request | The request contents are incorrectly formatted or a request is made for an invalid API version. | Error summary and optional message. |
| 401 Unauthorized | User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid. | None. |
| 403 Forbidden | The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted. | None. |
| 404 Not Found | The requested path does not refer to an existing object or object relation. | Error summary and optional message. |

| Response Code | Description | Response Body |
|--------------------|--|-------------------------------------|
| 406 Not Acceptable | The request is such that the Accept header does not indicate that JSON is an acceptable format for a response. | None. |
| 409 Conflict | The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists or an attempt was made to delete an object that is referred to by another object. | Error summary and optional message. |
| 500 Server Error | The server encountered an unexpected error. Please report server errors to customer support. | Error summary and optional message. |

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

```
{
  status: "Not Found"
  message: "Relation 'bad-relation' does not exist"
}
```

The Configuration extension has an `omit-error-message-details` property that suppresses the message in error responses, preventing the server from inadvertently revealing sensitive information. Set this property as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name "Configuration" \
  --set omit-error-message-details:true
```

Generating a Summary of Configuration Components

The Identity Proxy provides a `summarize-config` tool that generates a summary of the configuration in a local or remote identity data store instance. The tool is useful when comparing configuration settings on the identity data store 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 identity data store 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,
                          extended, 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 Identity Proxy 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 Identity Proxy 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 |
| 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 identity data store 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 identity data store 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 identity data store 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 identity data store 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 identity data store instance.

8. Test the server group by disabling the log publisher on the first identity data store 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 identity data store instance. The first identity data store 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 identity data store instance. Repeat this step on the third identity data store 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 Identity Proxy 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 Identity Proxy.

To limit full access to all of the Identity Proxy, 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 Identity Proxy 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. |

| Privilege | Description |
|-------------------|---|
| 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 Identity Proxy 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 Identity Proxy'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`. Whether the new root DN inherits the default set of root privileges is determined by the value of `ds-cfg-inherit-default-root-privileges`. Data under `cn=Root DNs,cn=config` is not replicated. Therefore the following steps need to be repeated on every system.

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

```
dn: cn=Data Admin,cn=Root DNs,cn=config
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
userPassword: password
cn: Data Admin
sn: Admin
ds-cfg-alternate-bind-dn: cn=Data Admin
givenName: Data
ds-cfg-inherit-default-root-privileges: false
ds-privilege-name: bypass-acl
ds-privilege-name: password-reset
ds-privilege-name: update-schema
ds-privilege-name: unindexed-search
ds-privilege-name: use-admin-session
ds-privilege-name: soft-delete-read
```

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

The Identity Proxy itself is also associated with a location. This location is specified in the global configuration properties of the Identity Proxy. If the load balancing algorithm's `use-location` property is set to `true`, then the load balancing component of the Identity Proxy refers to the Identity Proxy'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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy 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 Batched Transactions

You can configure the Identity Proxy to use batched transactions in both simple and entry-balanced configurations. The batched transactions feature supports two implementations: the standard LDAP transactions per RFC 5805 and the UnboundID proprietary implementation, known as the *multi-update extended operation*. Batched transactions can be used through the Identity Proxy in both simple and entry-balanced configurations although only in cases in which all operations within the transaction request may be processed within the same backend server and within the same Berkeley DB JE backend. Batched transactions cannot be processed across multiple servers or multiple Identity Data Store backends.

The multi-update extended operation makes it possible to submit multiple updates in a single request. These updates may be processed either as individual operations or as a single atomic unit. When the Identity Proxy receives a Start Batched Transaction request, it will queue all associated operations in memory until the End Batched Transaction request is received with the intention to commit, at which point the set of operations is sent as a single multi-update extended request to the Identity Data Store.

Add, delete, modify, modify DN, and password modify extended operations may be included in the set of operations processed during a batch transaction. The operations are processed sequentially in the order in which they were included in the extended request. If an error occurs while processing an operation in the set, then the server can be instructed to continue the processing or to cancel any remaining operations. If the operations are not cancelled, you can configure the server to process all operations as a single atomic unit.

Because of this use of multi-update, the external Identity Data Store must be configured to allow multi-update extended requests made by the Identity Proxy on behalf of the DN submitting the batched transaction. For example, the following Identity Data Store `dsconfig` command grants anonymous access to the multi-update extended request. The submitter of the request still needs access rights for the individual operations within the multiple-update.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(extop="1.3.6.1.4.1.30221.2.6.17")(version 3.0; acl "Anonymous
  access to multi-update extended request"; allow (read) userdn="ldap:///anyone");'
```

To Configure Batched Transactions

Batched transactions are managed by the Batched Transactions Extended Operation Handler. You can use it to configure the start transaction and end transaction operations used to indicate the set of add, delete, modify, modify DN, and/or password modify operations as a single atomic unit.

1. You can configure batched transactions using the `dsconfig` command as follows:

```
$ bin/dsconfig set-extended-operation-handler-prop \  
  --handler-name "Batched Transactions" \  
  --set enabled:true
```

2. Configure the external servers to allow the multi-update extended operation by granting access rights to the feature. See example in the previous section.

Configuring Server Health Checks

You can use the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy, see [About LDAP Health Checks](#).

About the Default Health Checks

By default, the Identity Proxy has two health check instances enabled for use on all servers:

- **Consume Admin Alerts.** This health check detects administrative alerts from the Identity Data Store, as soon as they are issued, by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Identity Data Store. When the Identity Proxy is notified by the Identity Data Store of a new alert, it immediately retrieves the base `cn=monitor` entry of the Identity Data Store. If this entry has a value for the `unavailable-alert-type` attribute, then the Identity Proxy will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Identity Proxy will consider it degraded.

- **Get Root DSE.** This health check detects if the root DSE entry exists on the LDAP external server. As this entry always exists on an UnboundID Identity Data Store, 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 Identity Proxy 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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy, 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 Identity Proxy 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 Identity Proxy to communicate with the server properly.

UnboundID Identity Proxy includes a tool, `prepare-external-server`, for configuring communication between the Identity Proxy and the LDAP backend server. After you add a new LDAP external server to an existing installation, we strongly recommend that you run this

tool to automatically create the user account necessary for communications. The `prepare-external-server` tool does not make configuration changes to the local Identity Proxy, only the external server is modified. When you run this tool, you must supply the user account and password that you specified for the Identity Proxy during configuration, `cn=Proxy User` by default.

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

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



About the `prepare-external-server` Tool

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

If you want the `prepare-external-server` tool to add the LDAP external server's certificates to the Identity Proxy's trust store, you must include the `--proxyTrustStorePath` option, and either the `--proxyTrustStorePassword` or the `--proxyTrustStorePasswordFile` option. The default location of the Identity Proxy 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 Identity Data Store on the remote host, `ds-east-01.example.com`, listening on port 1389 for access by the Identity Proxy 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 Identity Data Store to grant the proxy user the proxy access right.



Note: For non-UnboundID servers, the `--baseDN` argument is required for the `prepare-external-server` tool. The base DN is used to create the global ACI entries for these servers.

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

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

- Use the `prepare-external-server` tool to prepare the Identity Data Store. 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
```

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 Identity Proxy 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 Identity Data Store.

```
>>>> 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 Identity Proxy and the UnboundID Identity Data Store(s).

Configuring Load Balancing

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



Note: If the `use-location` property is set to true, then the load is balanced only among available external servers in the same location. If no external servers are available in the same location, the Identity Proxy 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 Identity Proxy to prefer local DEGRADED health servers to servers in a failover location. See the UnboundID Identity Proxy

Reference Guide for more information on the `prefer-degraded-servers-over-failover` property.

The Identity Proxy 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 Identity Proxy 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 Identity Proxy to use Criteria-Based Load-Balancing Algorithms to strike a balance between providing a consistent view of identity data store 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 Identity Proxy 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 Identity Proxy uses a Load-Balancing Algorithm to choose which Identity Data Store to use for a particular operation. The Load-Balancing Algorithm takes several factors into account when choosing a server:

- The availability of the identity data stores.
- The location of the identity data stores. By default Load-Balancing Algorithms prefer identity data stores in the same location as the Identity Proxy.
- Whether the Identity Data Store 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 Identity Data Stores 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 Identity Data Stores in the same location as the Identity Proxy. 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 Identity Proxy) 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 Identity Data Store 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 Identity Proxy'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 Identity Data Stores, 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 Identity Data Store instance than the first ADD operation, and this happens within the replication latency.
- Some clients add or modify an entry and then immediately read the entry back from the server, expecting to see the updates reflected in the entry.

In these situations, it is desirable to configure the `<keyword keyref="PROXY_SERVER_BASE_NAME"/>` to route dependent requests to the same server.

The server affinity feature (see [Configuring Server Affinity](#)) achieves this in some environments but not in all because the affinity is tracked independently by each Identity Proxy instance, and some clients send requests to multiple identity proxies. It is common for a client to not connect to the Identity Proxys directly but instead to connect through a network load balancer, which in turn opens connections to the Identity Proxys. Each individual client connection will be established to a single Identity Proxy so that operations on that connection will be routed to the same Identity Proxy, and server affinity configured within the Identity Proxy will ensure those operations will be routed to the same Identity Data Store. 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 Identity Proxys. Dependent operations sent on different connections could then be routed to different Identity Proxys, and then on to different Identity Data Stores.

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 Identity Data Store 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 Identity Proxy'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 Identity Data Store instances. This can reduce the WAN replication delay because the Identity Proxy will then prefer to send writes to the Identity Data Store with the WAN Gateway role, avoiding an extra hop to the remote locations.
- For Identity Proxy 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 an Identity Proxy'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 an Identity Proxy fronting three Identity Data Stores: `ds1.example.com`, `ds2.example.com`, and `ds3.example.com`.

Once these configurations changes are applied, the Identity Proxy 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:AAAvQvYsEavey80T0QfR60I \
--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 Set of Servers

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

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

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

The following procedure provides examples of each of these steps.

1. Create the new failover load balancing algorithm using `dsconfig` as follows:

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

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

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

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

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

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

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add criteria-based-load-balancing-algorithm: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 Identity Proxy 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 Identity Proxy 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 Identity Proxy does both proactive and reactive health checking. With proactive health checking, the Identity Proxy will periodically (by default, every 30 seconds), run a full set of tests against each backend server. The result of these tests will be used to determine the overall health check state (available, degraded, or unavailable) and score (an integer value from 10 to 0). With reactive health checking, the Identity Proxy 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 subsequent proactive health check determines that it has recovered. If a server is determined to be degraded, it may not become available until the next proactive health check, but it could be downgraded to unavailable by a reactive check if other failures are encountered against that server.

Both proactive and reactive health check assignments take effect immediately and will be considered for all subsequent requests routed to the load-balancing algorithm. If a 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy can perform extensive replacements, including attribute names used in DN's 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 Identity Proxy 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 Identity Proxy.
- **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 Identity Proxy.

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 Identity Proxy configuration console menu, enter the number associated with proxy transformations. On the Proxy Transformation management 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 Identity Proxy 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 identity data stores 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 Identity Proxy 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, subtrees, 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 Identity Proxy 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 Identity 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 Identity Proxy 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 Identity Proxy:

- **Client Connection.** Requests from the same Identity Proxy client connection are consistently routed to the same external server.
- **Client IP.** Identity Proxy 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 Identity Proxy 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 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 Identity Proxy 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 Identity Proxy'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 Identity Proxy 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 Identity Proxy.
- > 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 Identity Proxy 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 `£` to finish the creation of the client connection policy.

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

Configuring Globally Unique Attributes

The UnboundID Identity Proxy supports a Globally Unique Attributes feature that ensures uniqueness for any value defined for a set of attributes within a subtree view. You can also configure when the server checks for attribute conflicts, either prior to any add, modify, or modify DN change request (pre-commit) or after the successful completion of a change request (post-commit).

About the Globally Unique Attribute Plug-in

The Identity Proxy supports a Globally Unique Attribute Plug-in that prevents any value within a defined set of attributes to appear more than once in any entry for one or more subtree views. Administrators can also configure whether conflict validation should be checked before an add, modify, or modify DN request to one or more backend servers or after the change has successfully completed.

For example, if the `pre-commit-validation` property is enabled, the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected. If the `post-commit-validation` property is enabled, after the change has been processed, the server performs one or more searches to determine if a conflict was created in multiple servers at the same time. If a conflict is detected in this manner, then an administrative alert will be generated to notify administrators of the problem so that they can take any manual corrective action.



Note: The Globally Unique Attribute plug-in will attempt to detect and/or prevent unique attribute conflicts for changes processed through this Identity Proxy, but it cannot detect conflicts introduced by changes applied by clients communicating *directly* with backend servers.

We recommend that the Unique Attribute plug-in be enabled for all backend servers with the same configuration, so that conflicts can be detected within individual backend server instances. However, the Unique Attribute plug-in alone may not be sufficient for cases in which the content is split across

multiple sets of servers (e.g., in an entry-balanced environment or in proxy configurations with different branches on different servers).

In general, note the following points about pre-commit validation versus post-commit validation:

- Pre-commit validation is the only mechanism that can try to prevent conflicts. It will increase the processing time for add, modify, and modify DN operations because the necessary searches to look for conflicts happen before the update request is forwarded to any backend servers.
- Post-commit validation will only let you know (via administrative alert) about conflicts that already exist in the data. It can't prevent conflicts, but can allow you to deal with them in a timely manner. It also operates during the post-response phase, so it won't affect the processing time for the associated write operation.
- In most cases, pre-commit validation should be sufficient to prevent conflicts, although we recommend that you periodically run the `identify-unique-attribute-conflicts` tool to find any conflicts that may have arisen. If you want to mitigate any risks due to conflicts being generated by concurrent operations in different servers, then using both `pre-commit-validation` and `post-commit-validation` properties provides the best combination of preventing most conflicts in advance, and detecting and alerting about conflicts that arise from concurrent writes.

For more detailed information about the plug-in, see the *Identity Proxy Reference (HTML)*

To Configure the Globally Unique Attribute Plug-in

The following example shows how to configure the Globally Unique Attribute plug-in. The example defines an attribute set consisting of the `telephoneNumber` and `mobile` attributes within the "test-view" subtree view. The `multiple-attribute-behavior` property determines the scope of how attributes may differ among entries and is the same property for the Identity Data Store plug-in. The property is set to `unique-across-all-attributes-including-in-same-entry`, which indicates that the `telephone` and `mobile` attributes must be unique throughout the subtree view, even within an entry. The `pre-commit-validation` property ensures that the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected.

Note that all configured attributes should be indexed for equality in all backend servers.

- Run `dsconfig` to create the Globally Unique Attribute plug-in. The server will check that any add, modify, or modify DN request does not conflict with any attribute values in the entries. If there is a conflict, the change request will be rejected.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Globally-Unique telephone and mobile" \
  --type globally-unique-attribute \
  --set enabled:true \
  --set type:telephoneNumber \
  --set type:mobile \
  --set subtree-view:test-view \
  --set multiple-attribute-behavior:unique-across-all-attributes-including-in-same-entry \
```

```
--set pre-commit-validation:all-available-backend-servers
```

Configuring the Global Referential Integrity Plug-in

The UnboundID Identity Proxy supports a global referential integrity plug-in mechanism that maintains DN references from a specified set of attributes to entries that exist in the server (e.g., between the members values of a static group and the corresponding user entries). The plug-in intercepts delete and modify DN operations and updates any references to the target entry. For a delete operation, any references to the target entry are removed. For modify DN operations, any references to the target entry are updated to reflect the new DN of the entry.

The plug-in is similar to the Identity Data Store Referential Integrity Plug-in but does not have an asynchronous mode. When enabled on the Identity Proxy Server, the client response will be delayed until the referential integrity processing is complete. For Identity Proxy Server deployments not using entry balancing and using Identity Data Store external servers, it is best to instead use the Referential Integrity Plug-in on the Identity Data Store.

An equality index must be defined on all attributes referenced within the Global Referential Integrity Plug-in across all external servers.

Sample Global Referential Integrity Plug-in

- Use `dsconfig` to configure the Global Referential Integrity plug-in. The plug-in ensures that the `member`, `uniqueMember`, and `manager` attributes maintain their DN references in the defined subtree views. Note that any attributes for which referential integrity should be maintained should have values which are DNs and should be indexed for equality in all backend servers.

```
$ bin/dsconfig create-plugin \  
  --plugin-name "Global Referential Integrity" \  
  --type global-referential-integrity \  
  --set "enabled:true" \  
  --set "attribute-type:member" \  
  --set "attribute-type:uniqueMember" \  
  --set "attribute-type:manager" \  
  --set "subtree-view:employee-view" \  
  --set "subtree-view:groups-view"
```

Configuring an Active Directory Server Back-end

Configuring an Active Directory server back-end requires a `dsconfig` script. The following settings are required for an Active Directory server:

- `verify-credentials-method:bind-on-existing-connections`, and `authorization-method:rebind`

Active Directory does not support `proxy-as`. Existing connections must be reused.

- `set max-connection-age:5m`, and `health-check-pooled-connections:true`

Active Directory drops idle connections after 15 minutes. The proxy needs to refresh the connection pool in a shorter interval.

The following example `dsconfig` script configures two Active Directory servers (AD-SRV1 and AD-SRV2).

```
dsconfig set-ldap-health-check-prop --check-name "Consume Admin Alerts" \
--reset use-for-all-servers

dsconfig set-trust-manager-provider-prop \
--provider-name "Blind Trust" \
--set enabled:true

dsconfig create-external-server --server-name AD-SRV1 --type active-directory \
--set server-host-name:example.server \
--set server-port:636 \
--set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
--set password:password --set connection-security:ssl \
--set key-manager-provider:Null --set trust-manager-provider:"Blind Trust" \
--set authorization-method:rebind \
--set verify-credentials-method:bind-on-existing-connections \
--set max-connection-age:5m \
--set health-check-pooled-connections:true

dsconfig create-external-server --server-name AD-SRV2 --type active-directory \
--set server-host-name:example.server \
--set server-port:636 \
--set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
--set password:password \
--set connection-security:ssl \
--set key-manager-provider:Null \
--set trust-manager-provider:"Blind Trust" \
--set authorization-method:rebind \
--set verify-credentials-method:bind-on-existing-connections \
--set max-connection-age:5m \
--set health-check-pooled-connections:true

dsconfig create-load-balancing-algorithm --algorithm-name AD-LBA \
--type fewest-operations \
--set enabled:true \
--set backend-server:AD-SRV1 \
--set backend-server:AD-SRV2 \
--set use-location:false

dsconfig create-request-processor --processor-name AD-Proxy --type proxying \
--set load-balancing-algorithm:AD-LBA

dsconfig create-subtree-view --view-name AD-View \
--set base-dn:dc=dom-ad2,dc=local \
--set request-processor:AD-Proxy

dsconfig set-client-connection-policy-prop --policy-name default \
--set subtree-view:AD-View
```


Chapter

4

Managing Access Control

The UnboundID Identity Proxy 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 Identity Proxy.

Topics:

- [*Overview of Access Control*](#)
- [*Working with Targets*](#)
- [*Examples of Common Access Control Rules*](#)
- [*Validating ACIs Before Migrating Data*](#)
- [*Migrating ACIs from Sun/Oracle to UnboundID Identity Data Store*](#)
- [*Working with Privileges*](#)

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources.** *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name.** *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients.** *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights.** *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The UnboundID Identity Proxy provides important access control features that provide added security for the Identity Proxy's entries.

Improved Validation and Security

The Identity Proxy provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Identity Proxy 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 Identity Proxy examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to

an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the UnboundID Identity Proxy, 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 Identity Proxy include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.
- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

Access Controls for Public or Private Backends

The UnboundID Identity Proxy classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Identity Proxy itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The UnboundID Identity Proxy access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Identity Proxy iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissible operations. When a client application, such as `ldapsearch`, enters a request, the Identity Proxy 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.

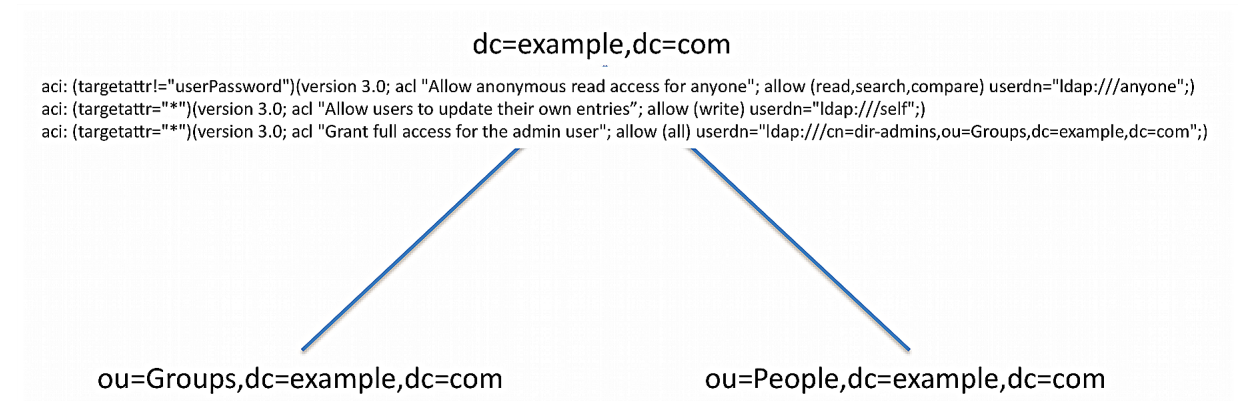


Figure 3: ACI

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: <i>(target keyword = != expression)</i> |
| name | Specifies the name of the ACI. |
| permissions | Specifies the type of operations to which an access control rule might apply. Syntax: <i>allow deny (permission)</i> |
| bind rules | Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: <i>bind rule keyword = != expression</i> ; 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 Identity Proxy 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 if 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. |

| Permission | Description |
|------------|---|
| 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 <code>export</code> permission for operations against the entry's original DN. The requestor must have the <code>import</code> 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 <code>write</code> 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 <code>export</code> permission. |
| proxy | Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identity 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 | <p>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:</p> <pre>authmethod = <i>method</i></pre> <p>where <i>method</i> is one of the following representations:</p> <ul style="list-style-type: none"> > 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. <p>The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (userdn="ldap:///self" and authmethod="ssl");)</pre> |
| dayofweek | <p>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:</p> <pre>dayofweek = <i>day1</i>, <i>day2</i>, ...</pre> <p>where <i>day</i> is one of the following representations:</p> <ul style="list-style-type: none"> > sun > mon > tues > wed > thu > fri > sat <p>The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:</p> <pre>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");)</pre> |
| dns | <p>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:</p> <pre>dns = <i>dns-host-name</i></pre> <p>The following example allows users on hostname <code>server.example.com</code> to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (dns="server.example.com" and userdn="ldap:///self");)</pre> |

| Bind Rule Keyword | Description |
|-------------------|---|
| groupdn | <p>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.</p> <pre>groupdn [= !=] "ldap:///groupdn [ldap:///groupdn] ..."</pre> <p>The following example allows users in the managers group to update their own entries:</p> <pre>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");)</pre> |
| ip | <p>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:</p> <pre>ip [= !=] ipAddressList</pre> <p>where <i>ipAddressList</i> is one of the following representations:</p> <ul style="list-style-type: none"> ➤ 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. <p>The following example allows users on 10.130.10.2 and localhost to update their own entries:</p> <pre>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");)</pre> |
| timeofday | <p>Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:</p> <pre>timeofday [= != >= > <= <] time</pre> <p>where <i>time</i> is one of the following representations:</p> <ul style="list-style-type: none"> ➤ 4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour) ➤ Wildcards are not allowed in this expression <p>The following example allows users to update their own entries if the request is received before 12 noon.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users who authenticate before noon to update their own entries"; allow (write) (timeofday<1200 and userdn="ldap:///self" and authmethod="simple");)</pre> |
| userattr | <p>Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of 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</p> |

| Bind Rule Keyword | Description |
|-------------------|---|
| | <p>other than USERDN, GROUPODN, or LDAPURL is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:</p> <pre>userattr = attrName# [bindType attrValue]</pre> <p>where:</p> <ul style="list-style-type: none"> ➤ <i>attrName</i> = name of the attribute for matching ➤ <i>bindType</i> = USERDN, GROUPODN, LDAPURL ➤ <i>attrValue</i> = an attribute value. Note that the <i>attrVALUE</i> of the attribute must match on both the bind entry and the target of the ACI. <p>The following example allows a manager to change employee's entries. If the bind DN is specified in the <i>manager</i> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p> <pre>aci: (targetattr="") (version 3.0; acl "Allow a manager to change employee entries"; allow (write) userattr="manager#USERDN");</pre> <p>The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the <i>allowEditors</i> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p> <pre>aci: (targetattr="") (version 3.0; acl "Allow allowEditors to change employee entries"; allow (write) userattr="allowEditors#GROUPODN");</pre> <p>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.</p> <pre>aci: (targetattr="") (version 3.0; acl "Allow managers to change employees entries two levels below"; allow (write) userattr="parent[0,1,2].manager#USERDN");</pre> <p>The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.</p> <pre>aci: (targetattr="") (version 3.0; acl "Allow any member of Eng Dept to update any other member of the engineering department at or below the ACI"; allow (write) userattr="department#ENGINEERING");</pre> <p>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 <i>allowedEditorCriteria</i> attribute of the target entry.</p> <pre>aci: (targetattr="") (version 3.0; acl "Allow a user that matches the filter to change entries"; allow (write) userattr="allowedEditorCriteria#LDAPURL");</pre> |
| userdn | <p>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:</p> <pre>userdn [= !=] "ldap:///value ["ldap:///value ..."]</pre> <p>where <i>value</i> is one of the following representations:</p> <ul style="list-style-type: none"> ➤ The DN of the target user ➤ A value of <i>anyone</i> to match any client, including unauthenticated clients. ➤ A value of <i>all</i> to match any authenticated client. |

| Bind Rule Keyword | Description |
|-------------------|--|
| | <ul style="list-style-type: none"> > A value of <code>parent</code> to match the client authenticated as the user defined in the immediate parent of the target entry. > A value of <code>self</code> to match the client authenticated as the user defined in the target entry. <p>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.</p> <p>The following example allows users to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) userdn="ldap:///self";)</pre> |

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targetattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (`!=`) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following

target expression matches all entries that contains and begins with "john.d," so that entries like "john.doe,ou=People,dc=example,dc=com," and "john.davies,ou=People,dc=example,dc=com" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the ou attribute. Entries like "john.doe,ou=People,dc=example,dc=com," and "john.davies,ou=asia-branch,dc=example,dc=com" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the ou=People,dc=example,dc=com branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
(targetattr="*")
(version 3.0; acl "Allow all the ou=People branch to modify their own entries";
allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the UnboundID Identity Proxy:

- **(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 overridden default constraints like size limit, time limit, or look-through limit for that user.

The UnboundID Identity Proxy distinguishes between these two types of attributes in its access control implementation. The Identity Proxy 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 Identity Proxy 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 an Identity Proxy 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 Identity Proxy does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn|sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribute:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing))")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
(targetattr="departmentNumber|cn|sn")
(version 3.0; acl "example"; allow (write)
 userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

- > **add** represents the operation of adding an attribute value to the entry

- **del** represents the operation of removing an attribute value from the entry
- **attr1, attr2... attrN** represents the targeted attributes
- **filter1, filter2 ... filterN** represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=identity data store admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-read)")
(version 3.0; acl "Allow members of the identity data store admins group to grant the
soft-delete-read privilege"; allow (write)
groupdn="ldap:///cn=identity data store admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is `base` to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
(targetattr="supportedControl||supportedExtension||
supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
(version 3.0; acl "Allow users to view Root DSE Operational Attributes";)
```

```
allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
(version 3.0; acl "Allow admins to use the Soft Delete Request Control,
Hard Delete Request Control,Undelete Request Control, and
Soft-deleted entry access request control";
allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
(version 3.0; acl "Allows the mgr to use the Password Modify Request and StartTLS;
allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the `"cn=admins,ou=groups,dc=example,dc=com"` group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or compare
operational attributes";
allow (read,search,compare) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

```
aci: (targetattr="*)(version 3.0; acl "Administrators can add, modify and delete
entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries representing
people";
allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries representing
people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can change
password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the data store, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to `false`.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application `"cn=OnBehalf,ou=applications,dc=example,dc=com"` to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the UnboundID Identity Proxy rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to an UnboundID Identity Proxy.

To validate an access control instruction, the UnboundID Identity Proxy provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Windows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
    (version 3.0; acl "Allow anonymous read access for anyone";
      allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
    (version 3.0; acl "Allow users to update their own entries";
      allow (write) userdn="ldap:///self";)

# The following access control rule is valid dn:
dc=example,dc=com
```

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

3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif \
  --onlyReportErrors
```

```
# Processing complete
# Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

The output file shows the following message:

```
# The following access control rule is malformed or contains an unsupported
# syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
# "Allow anonymous read access for anyone"; allow (read,search,compare)
# userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
# Instruction (ACI) because it failed general ACI syntax evaluation
dn: dc=example,dc=com
aci: (targetattr!=userPassword)
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in Another Identity Proxy

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Identity Proxy that you are planning to migrate to the UnboundID Identity Proxy. The tool helps to determine whether the UnboundID Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Identity Proxy, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to UnboundID Identity Data Store

This section describes the most important differences in access control evaluation between Sun/Oracle and the UnboundID Identity Data Store.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the UnboundID Identity Data Store. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The UnboundID Identity Data Store does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the UnboundID Identity Data Store, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except `userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName|modifiersName"` . All operational attributes can be targeted using the `+` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*|+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and UnboundID Identity Data Store support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the UnboundID Identity Data Store, global ACIs are managed with `dsconfig` referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the UnboundID Identity Data Store, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as `"oid={oid},cn=features,cn=config"` where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: `"oid=2.16.840.1.113730.3.4.3,cn=features,cn=config"`.

In the UnboundID Identity Data Store, the `"targetcontrol"` keyword can be used to define ACIs that grant or deny access to controls. The `"extop"` keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The UnboundID Identity Data Store is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.



Note: Consider running the `import-ldif` tool with the `--rejectFile` option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The UnboundID Identity Data Store offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The UnboundID Identity Data Store provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the UnboundID Identity Data Store instance. When migrating data from a DSEE deployment into an UnboundID Identity Data Store instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the UnboundID Identity Data Store.

For more information about using this tool, see [Validating ACIs Before Migrating Data](#).

Working with Privileges

In addition to the access control implementation, the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy.

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 <code>audit-data-security</code> tool. |
| backend-backup | This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend. |
| backend-restore | This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend. |
| bypass-acl | This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the <code>bypass-acl</code> privilege but without the <code>config-read</code> 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 <code>config-read</code> 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 Identity Proxy. The user is also required to have the <code>jmx-read</code> privilege. |
| jmx-read | This privilege is required for a user to access any information provided by the Identity Proxy via the Java Management Extensions (JMX). |
| jmx-write | This privilege is required for a user to update any information exposed by the Identity Proxy via the Java Management Extensions (JMX). The user is also required to have the |

| Privilege | Description |
|------------------|---|
| | jmx-read privilege. Note that currently all of the information exposed by the server over JMX is read-only. |
| ldif-export | This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group. |
| ldif-import | This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the ldif-export privilege. |
| lockdown-mode | This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode. |
| modify-acl | This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the aci operational attribute. |
| password-reset | This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the userPassword attribute to the target entry. |
| privilege-change | This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the ds-privilege-name operational attribute. |
| 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 Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
--add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*|+")(
(version 5.0; acl "Access to the tasks backend for administrators";
allow (all) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com"; )'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the UnboundID Identity Proxy is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege

(for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Chapter

5

Deploying a Standard Identity Proxy

You can deploy UnboundID Identity Proxy in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates a standard deployment scenario.

Topics:

- [*Creating a Standard Multi-Location Deployment*](#)
- [*Expanding the Deployment*](#)
- [*Merging Two Data Sets Using Proxy Transformations*](#)

Creating a Standard Multi-Location Deployment

In this example deployment, UnboundID Identity Proxy will be deployed in the data centers of two geographic locations: east and west. All LDAP external servers in this deployment are UnboundID Identity Data Stores. 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 UnboundID Identity Data Store instances in two locations with replication of the `dc=example,dc=com` base DN enabled:

- > ds-east-01.example.com
- > ds-east-02.example.com
- > ds-west-01.example.com
- > ds-west-02.example.com

We will configure four Identity Proxy instances:

- > proxy-east-01.example.com
- > proxy-east-02.example.com
- > proxy-west-01.example.com
- > proxy-west-02.example.com

Overview of the Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the first Identity Proxy 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 identity proxy in the east location using the `setup` or `setup.bat` file included in the zip installation file and copy the configuration of the first Identity Proxy using the configuration cloning feature.
- Install two Identity Proxy instances in the west location, which includes using the `setup` file and manually setting the location to west using the `dsconfig` command, as well as copying the configuration of the Identity Proxy using the configuration cloning feature.

After the identity proxy has been configured and tested, we then provide a tour of the configuration of each of the identity proxy components. These properties can be modified later as needed using the `dsconfig` tool.

Installing the First Identity Proxy

To begin with, we have the UnboundID Identity Proxy installation zip file. In this example, we plan to use SSL security, so we also have a keystore certificate database and a pin file that contains the private key password for the keystore. The keystore files are only necessary when using SSL or StartTLS.

In this deployment scenario, the keystore database is assumed to be a Java KeyStore (JKS), which can be created by the keytool program. For more information about using the keytool, see the "Security Chapter" in the UnboundID Identity Data Store Administration Guide.

The UnboundID-Proxy directory contains the following:

```
root@proxy-east-01: ls
ExampleKeystore.jks  ExampleTruststore.jks ExampleKeystore.pin
UnboundID-Proxy-5.0.1.0-with-je.zip
```

The `ExampleKeystore.jks` keystore file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, CA, and intermediate signing certificates are all contained in the `ExampleTruststore.jks` file. The password for `ExampleKeystore.jks` is defined in clear text in the corresponding pin file, though the name of the file need not match as it does in our example. The private key password in our example is the same as the password defined for the `ExampleKeystore.jks` keystore.

To Install the First Identity Proxy

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 identity proxy 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 Identity Proxy, you can configure it using the `create-initial-proxy-config` tool as presented in the next section.

Configuring the First Identity Proxy

Once the Identity Proxy 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 identity proxy configuration.

Configuring the Identity Proxy with the `create-initial-proxy-config` tool involves the following steps:

- Providing an Identity Proxy 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 Identity Proxy.

To Configure the First Identity Proxy

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 Identity Proxy will use to authenticate to the backend UnboundID Identity Data Store instances. The `create-initial-proxy-config` tool requires that the same bind DN and password be used to authenticate to all of the backend servers. All of our Identity Proxy instances have identical proxy user accounts and passwords. If necessary, the proxy user account password can be defined differently for each external server using `dsconfig` after the `create-initial-proxy-config` tool has been executed.

```
>>>> >>>> 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 UnboundID Identity Data Store instances. 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 DN of the UnboundID Identity Data Store instances that the Identity Proxy will access. For this example, use `dc=example,dc=com`.

```
>>>> >>>> Proxy Base DNs

Enter the 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]:

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 DN of the UnboundID Identity Data Store instances that will be accessed through the identity proxy. Because we are only using one proxy base DN, press **Enter** in this example.

```
>>>> >>>> Proxy Base DNs

Enter the 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]:
```

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 Identity Proxy. 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 Identity Proxy itself. Our Identity Proxy 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 identity data stores. 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 Identity Proxy.

```
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 Identity Proxy

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 Identity Proxy

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 Identity Proxy. 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 Identity Proxy. 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 identity proxy, 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 identity proxy 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 Identity Proxy Instances

We install and configure the second Identity Proxy by running the `setup` tool on `proxy-east-02.example.com`.

To Configure Additional Identity Proxy Instances

1. Copy the keystore and pin files into the `config` directory for the `proxy-east-02.example.com` server.

```
root@proxy-east-02: cp ../Keystore* config/
root@proxy-east-02: cp ../Truststore* config/
```

2. Install the second Identity Proxy 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 Identity Proxy <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 Identity Proxy, `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 Identity Proxy <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 Identity Proxy 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 Identity Proxy <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 identity proxies have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the Identity Proxy Global Configuration object. When making a change to an Identity Proxy using the `dsconfig` command-line tool or the Web console, you will have the choice to apply the changes locally only or to all identity proxies in the `all-servers` group.

Testing External Server Communications After Initial Setup

After setting up the basic deployment scenario, the communication between the identity proxies and the LDAP external servers can be tested using a feature in the identity proxy in combination with an LDAP search.

To Test the External Communications After Initial Setup

After initial setup, the Identity Proxy 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 Identity Proxy with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the Identity 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 Identity Proxy 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 Identity 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 identity 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 Identity Data Store 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 Identity Proxy instance on ds-east-01.example.com. Check the access log to confirm that the Identity Proxy 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 Identity Data Store 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 identity proxies will use StartTLS to communicate with the directory servers in the central region.



Note: Other than the ability to add to the Identity Proxy's truststore, the `prepare-external-server` tool does not alter the Identity Proxy configuration in any way.

The Identity Proxy itself, installed on proxy-east-01.example.com, remains in the East location. This example will reconfigure round-robin load balancing between the six identity data stores 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 identity data stores, `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` UnboundID Identity Data Store 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 Identity Proxy 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 Identity Proxy.

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 Identity Proxy 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 modify 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
```



Note: For entry-balancing deployments, the global base DN is required when invoking prepare-external-server.

Adding the New UnboundID Identity Data Stores to the Identity Proxy

After preparing the external UnboundID Identity Data Stores to communicate with the Identity Proxy, we can now add the two servers in the central location to the identity proxy instance. Because we have run the prepare-external-server tool, the two servers have the cn=Proxy User entry configured.

To Add the New UnboundID Identity Data Stores to the Identity 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 UnboundID Identity Data Stores 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 Configuration Console main menu, enter the number corresponding to **Location**.

```
>>>> UnboundID Identity Proxy 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 number 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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' property
>>>> 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 Identity Proxy 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 Identity Proxy. The load-balancing algorithm is referenced in the load-balancing-algorithm property of the request processor used by this Identity Proxy.

7. To view this property, go to the main configuration console menu and select “Request Processor”.

```
>>>> UnboundID Identity Proxy 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 Identity Proxy. 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,
```

```
8) supported-control-oid      virtual-attributes-only
                                -
?) 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 Identity Proxy, which is in turn referenced by the client connection policy.



Note: The changes made in this procedure are already in effect. The Identity Proxy does not have to be restarted.

Testing External Server Communication

After adding and configuring the new external servers, test the communication between the Identity Proxy and the LDAP external servers using the `include-backend-server-passthrough-subtree-views` property of the Identity Proxy 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 `ldapsearch` 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 Identity Proxy 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 Identity 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 Identity Data Store 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 Identity Data Store 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 Identity Data Store 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 |
|-------------------------------|--------------------------------|
| <code>ou=east,o=sample</code> | <code>dc=example,dc=com</code> |
| <code>ou=west,o=sample</code> | <code>dc=example,dc=com</code> |
| <code>o=sample</code> | <code>dc=example,dc=com</code> |

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 Identity Data Store'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 Identity Proxy 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 Identity Proxy.
- 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 Identity Proxy inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at Identity Proxy 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 Identity Proxy. The following schema definitions are assumed to exist on the external Identity Data Store:

```
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 server root 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 Configuration Console main menu, enter the number corresponding to **Proxy Transformation**.

```
>>>> UnboundID Identity Proxy 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=example,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-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 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 transformation `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 example.

```
>>>> 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-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 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 example.

```
>>>> 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 Configuration Console main menu, enter the number corresponding to **Request Processor**.

```
>>>> UnboundID Identity Proxy 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

```

- 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

```

- 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

```

- 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

```

- 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

```

- 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 Identity Proxy.

To Create Subtree Views

1. On the Configuration Console main menu, enter the number corresponding to **Subtree View**.

```
>>>> UnboundID Identity Proxy 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 Configuration Console main menu, enter the number corresponding to **Client Connection Policy**.

```
>>>> UnboundID Identity Proxy 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 Identity Proxy will now respond to requests to the `dc=example,dc=com` and `o=sample` base DN's. 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}A504RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZX==
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}A504RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZX==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
exSampleRegion: west
exSampleLinkedAccounts: uid=jcase,ou=People,dc=example,dc=com
```


Chapter

6

Deploying an Entry-Balancing Identity Proxy

You can deploy UnboundID Identity Proxy in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates an entry-balancing deployment scenario.

Topics:

- [*Deploying an Entry-Balancing Proxy Configuration*](#)
- [*Rebalancing Your Entries*](#)
- [*Managing the Global Indexes in Entry-Balancing Configurations*](#)
- [*Working with Alternate Authorization Identities*](#)

Deploying an Entry-Balancing Proxy Configuration

Entry-balancing is an Identity Proxy 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 identity data stores, making the Identity Proxy 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 Identity Proxy. Replication on the external identity data stores must be properly configured before proceeding through this example. The identity data stores 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 Identity Proxy communicates with four external identity data stores. The Identity Proxy 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 Identity Data Store external servers describes the replication configuration that exists before creating the Identity Proxy configuration.

```
--- Replication Status for dc=example,dc=com: Enabled ---
Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
-----:-----:-----:-----:-----:-----
austin-set1.example.com:389 : 10003 : 0 : N/A : 722087263
austin-set2.example.com:389 : 10003 : 0 : N/A : 722087263
newyork-set1.example.com:389 : 10003 : 0 : N/A : 722087263
newyork-set2.example.com:389 : 10003 : 0 : N/A : 722087263

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset1): Enabled ---
Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
-----:-----:-----:-----:-----:-----
austin-set1.example.com:389 : 100001 : 0 : N/A : 178892712
newyork-set1.example.com:389 : 100001 : 0 : N/A : 178892712

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset2): Enabled ---
Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
-----:-----:-----:-----:-----:-----
austin-set2.example.com:389 : 100001 : 0 : N/A : 1057593890
newyork-set2.example.com:389 : 100001 : 0 : N/A : 1057593890
```

Determining How to Balance Your Data

If a single Identity Data Store 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 Identity Data Store instance can hold all of the information within one or more branches. Configure the Identity Proxy 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 identity data store instance as their parent.
- If one or more branches are so large that any single Identity Data Store 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 Identity Data Stores. 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 identity data stores outside the balancing point. This data is referred to as the global domain. The Identity Proxy entry-balancing configuration will contain at least two subtree views and associated request processors, one for the global domain and one for the entry-balancing domain. In our examples, the global domain is `dc=example,dc=com` and the entry-balancing domain is `ou=people, dc=example,dc=com`. The entry-balancing base DN, `ou=people,dc=example,dc=com`, is also the balancing point.

Entry Balancing and ACIs

In an entry-balancing deployment, access control instructions (ACIs) are still configured in the backend Identity Data Store data. When defining access controls in an entry-balancing deployment, you need to ensure that the data used by the access control rule is available for evaluation on all datasets.

If you use groups for access control and a group contains users from different data sets, then that group must exist on each dataset. For a single ACI to be applicable to entries in all datasets, it must be specified above the entry-balancing point. For example, if an ACI allows access to modify users that are part of group 1, then two things must exist on both data sets:

- Group 1 must exist in the `ou=groups` branch of both datasets.

- The ACI referencing group 1 must exist in the `ou=people` branch or above. The `ou=people` branch entry itself is part of the common data.

The Identity Proxy ensures that any changes to entries within the scope of the entry-balancing request processor, but outside the balancing point, are applied to all backend server sets. Any ACI stored at the entry-balancing point will be kept in sync if changes are made through the Identity Proxy.

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the Identity Proxy on `austin-proxy1`.
- Use the `create-initial-proxy-config` tool to provide our initial setup for entry balancing. The initial setup includes defining multiple subtree views and global indexes in support of entry balancing.
- Change the placement algorithm of the `austin-proxy-01` server to use an entry-count placement algorithm. This algorithm is used to select the backend set to which to forward an add request. It looks at the number of entries in the backend sets and forwards the add request to the backend with either the fewest or the most entries, depending on the configuration. You can also configure the placement algorithm to make the decision based on the on-disk database size rather than the number of entries.

Installing the Identity Proxy

We start by configuring the Identity Proxy. 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 Identity Proxy

- 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 Identity Proxy

Once the Identity Proxy 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 Identity Proxy configuration.

To Configure the Entry-Balancing Identity Proxy

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 identity proxies have identical proxy user accounts and passwords.

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

```
Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

```
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify the type of security 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 Identity Proxy 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 DN.

```

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 Identity Proxy 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 Identity Proxy, 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 Identity Proxy.

```
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 Identity Proxy.

```

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

```

```
How do you want to connect to the Directory Proxy Server on localhost?

1) LDAP
2) LDAP with SSL
3) LDAP with StartTLS

Enter choice [1]:

Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
Creating Locations ..... Done
Updating Failover Locations ..... Done
Updating Global Configuration ..... Done
Creating Health Checks ..... Done
Creating External Servers ..... Done
Creating Load-Balancing Algorithm for dc=example,dc=com .... Done
Creating Request Processor for dc=example,dc=com ..... Done
Creating Subtree View for dc=example,dc=com ..... Done
Updating Client Connection Policy for dc=example,dc=com ..... Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server Set 1 .....
Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set 1...Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server Set 2 ....
Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set 2...Done
Creating Entry Balancing Request Processor for ou=people,dc=example,dc=com ..... Done
Creating Placement Algorithm for ou=people,dc=example,dc=com .... Done
Creating Global Attribute Indexes for ou=people,dc=example,dc=com ..... Done
Creating Subtree View for ou=people,dc=example,dc=com ..... Done
Updating Client Connection Policy for ou=people,dc=example,dc=com ..... Done

See /logs/create-initial-proxy-config.log for a detailed log of this operation

To see basic server configuration status and configuration you can launch /bin/status
```

Configuring the Placement Algorithm Using a Batch File

Now, we configure the placement algorithm using a batch file. We want to place new entries added through the proxy via LDAP ADD operations into the least used dataset. We do this using an entry-count placement algorithm. To change the placement algorithm from round robin to entry count, we first create and enable an entry-count placement algorithm configuration object and then disable the existing round robin placement algorithm. Our batch file, `dsconfig.post-setup`, contains the `dsconfig` commands required to create the entry-count placement algorithm and disable the old round robin algorithm.

To Configure the Placement Algorithm Using a Batch File

The batch file contains comments to explain each `dsconfig` command. Note that in this example, line wrapping is used for clarity. The `dsconfig` command requires that the full command be provided on a single line.

The batch file itself looks like the following:

```
root@austin-proxy1:more ../dsconfig.post-setup

# This dsconfig operation creates the entry-count placement
# algorithm with the default behavior of adding entries to the
# smallest backend dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

# Note that once the entry-count placement algorithm is created
# and enabled, we can disable the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
```

```
# algorithm, we add a second algorithm and then disable the
# original round-robin algorithm created during the setup
# procedure.

dsconfig set-placement-algorithm-prop
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false

# At this point, LDAP ADD operations will be forwarded to an external
# server representing the dataset with the least number of entries.
```

- Run the `dsconfig` command using the batch file. Once the batch file has executed, a new entry-count placement algorithm, called entry-count, has been created, and the old round-robin placement algorithm, round-robin, has been disabled.

```
root@austin-proxy1: bin/dsconfig --no-prompt \
--bindDN "cn=directory manager" --bindPassword password \
--port 389 --batch-file ../dsconfig.post-setup
```

Batch file '`../dsconfig.post-setup`' contains 2 commands

```
Executing: create-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

Executing: delete-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false
```

Rebalancing Your Entries

If your deployment distributes entries using an entry counter placement algorithm or 3rd party algorithm, you may need to redistribute your entries. For example, imagine that you have an environment that distributes entries across three backends using an entry counter placement algorithm. This algorithm distributes entries to the backend that has the most space. Imagine that the backends all reach their maximum capacity and you decide to add a new backend to the deployment. You need to move the entries from the full backends and distribute them evenly across all the backends, including the new backend.

You might also want to deliberately rebalance your entries to meet the needs of your organization. For example, you can direct entry balancing based on attributes on the entries themselves. You can write a custom algorithm that looks at the value of an attribute that is being modified on the entry. Based on the attribute, you can then put this entry somewhere specific. You might use this feature if you want to have certain entries closer geographically to the client application using them. The geographical information could be included in the entry. Rebalancing would be used to move these entries to the server in the correct geographical location.

You can redistribute entry-balanced entries in two ways:

- **Using dynamic rebalancing.** With dynamic rebalancing, as existing entries get modified, they get moved. You configure dynamic rebalancing in the entry counter placement algorithm.

- **Using the `move-subtree` tool.** This tool can be used to move either small subtrees through a transactional method or to move large subtrees, potentially taking them offline for a short period.

The remainder of this section describes each of these method of entry rebalancing in more detail.

About Dynamic Rebalancing

During dynamic rebalancing entries get moved as they are modified. You configure dynamic rebalancing on the entry counter placement algorithm or a third-party placement algorithm that supports rebalancing. This algorithm keeps a count of the number of entries or the size of the backend set. You configure dynamic rebalancing using the following parameters:

- **`rebalancing-enabled`.** Determines whether entry rebalancing is enabled. When rebalancing is enabled, the placement algorithm is consulted after modify and add operations, to determine whether the target entry should be moved to a different backend set.
- **`rebalancing-scope`.** Indicates which modified entries are candidates for rebalancing. A value of `top-level` indicates that only entries immediately below the entry-balancing base can be rebalanced. A value of `any` indicates that entries at any level below the entry-balancing base may be rebalanced.
- **`rebalancing-minimum-percentage`.** Specifies the minimum threshold for entries to be migrated from one backend set to a preferred backend set with a smaller size. Entries are not migrated unless the percentage difference between the value of the current backend set and the value of the preferred backend set exceeds this threshold. This parameter prevents unnecessarily migrating entries back and forth between backend sets of similar sizes.
- **`rebalancing-subtree-size-limit`.** Specifies the maximum size of a subtree that can be rebalanced.
- **`poll-interval`.** Specifies how long to wait between polling the size of the backends to determine how to rebalance; works in conjunction with the `rebalancing-minimum-percentage` property.
- **`placement-criteria`.** Determines which approach to use to select a destination backend for rebalancing. Possible values are: `entry-count`, `backend-size`, or `custom`.

The following figure illustrates an entry-balancing base DN and three subtrees, A, B, and C. If the rebalancing scope is set to `any`, any child entries under the base DN can be rebalanced. For example, if a change is made to entry A1, the entire subtree A might be rebalanced, depending upon how you have configured rebalancing. If the rebalancing scope is set to `top-level`, rebalancing is only triggered when entries at the top level, such as A, are modified. Changes made to subentries, such as A1 or A2, do not trigger rebalancing. Rebalancing is also triggered upon the addition of entries such as A1, A2, provided the scope is `any`.

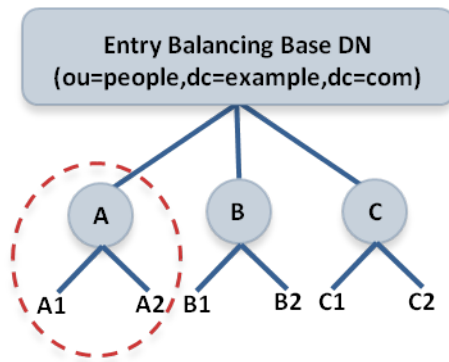


Figure 4: Rebalancing at the Top Level

If you are writing your own 3rd party algorithm, you program dynamic rebalancing using the `SelectRebalancingBackendSet` method on the placement algorithm. For more information, see the Server SDK documentation.

To Configure Dynamic Rebalancing

This procedure describes how to configure dynamic rebalancing on an existing entry balancing configuration.

1. To configure entry rebalancing, you may create an entry counter placement algorithm, if the current placement algorithm does not support rebalancing. You can either do this using `dsconfig` in interactive mode, or using the `dsconfig` command line as follows:

```
$ dsconfig create-placement-algorithm \
  --processor-name dc_example_dc_com-eb-req-processor \
  --algorithm-name rebalancing --type entry-counter \
  --set enabled:true --set rebalancing-enabled:true
```

2. Remove any placement algorithm previously configured on this entry-balancing request processor.
3. You can throttle how many entries are being moved by the proxy, so that the backend servers do not have too heavy a load. To do this, set the `rebalancing-queue-maximum-size` property of the request processor created in the previous step. By default, it is set to 1000. If the load is too high, reduce this value as follows:

```
$ dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set rebalancing-queue-maximum-size:50
```

4. Verify that the access logs are configured to display the subtrees being moved by dynamic rebalancing. The access logs provide a good way to monitor progress. So, if the write load on the backend servers is high and you see lots of rebalancing activity in the access log, lower the queue size to lower the rebalancing activity. You can configure the access log to display entry rebalancing processing information as follows:

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set log-entry-rebalancing-requests:true
```

About the move-subtree Tool

The `move-subtree` tool allows you to specify subtrees for rebalancing. You specify the source server, the target server, and one or more base DN's 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 DN's of the subtrees you want to move.

Note: The `move-subtree` tool requires users to have access to the extended operations and controls needed to run the tool. Make sure to apply the following ACIs to your data.



```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.5 ||  
      1.3.6.1.4.1.30221.2.5.24 || 1.3.6.1.4.1.30221.2.5.13")  
      (version 3.0; acl "Allow admin to submit move-subtree controls";  
        allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)  
aci: (extop="1.3.6.1.4.1.30221.2.6.19")  
      (version 3.0; acl "Allow admin to request move-subtree extended  
        operation"; allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

About the subtree-accessibility Tool

The `subtree-accessibility` tool helps you determine if a subtree has restricted access and helps you fix any problems. If, during rebalancing, the Identity Data Store issues an alert that a subtree has been unavailable for too long, then you can use this tool to evaluate the problem. For example, if the `move-subtree` tool is interrupted by a host machine going down unexpectedly, the subtree might not be successfully moved. You can use the `subtree-accessibility` tool to evaluate and correct any problems with the subtrees, and then re-run the `move-subtree` tool.

Managing the Global Indexes in Entry-Balancing Configurations

In an entry-balancing configuration, the Identity Proxy maintains the default RDN index as well as one or more in-memory global attribute indexes. The global indexes allow the Identity Proxy 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 identity data stores when the server starts up, and are updated by certain operations that come through the Identity Proxy. For instance, when a new entry is added, the DN of the new entry is added to the DN index of the Identity Proxy performing the operation. The indexes are also fault-tolerant and can adapt to

changes made in the backend servers without going through the Identity Proxy. 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 Identity 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 Identity Proxy 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 Identity 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 an Identity 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 Identity Proxy.

```
$ 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 Identity 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 Identity Proxy provides a tool, `reload-index`, which allows you to manually reload the Identity Proxy global indexes. You might need to reload the index when:

- The Identity 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 Identity Proxy'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 Identity Proxy to prime from the backend identity data store 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 identity proxies do not communicate changes to the indexes with one another. The Identity 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 Identity Proxy 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 Identity Proxy.

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 identity data stores. 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 Identity Proxy 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 Identity Data Store 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 Identity Data Store. 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 Identity Proxy can prime the global indexes on startup from the backend identity data store or from a peer identity proxy, 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 identity data stores in the topology. You can specify the data sources for the index priming and the order in which priming from these sources occurs.

Use the `prime-index-source` property to specify the sources of data, either `ds`, `file` or some combination of the two. The order you specify is the order in which priming from these sources will be attempted. For example, if you specify `prime-index-source:file,ds`, priming will be performed from the `global-index` data file created from the previous run of the identity data stores. With the `file,ds` configuration, the contents of the global index are written to disk periodically if, and only if, the entire global index has been primed previously from a identity data stores source either from startup or `reloaded-index`. Priming is most efficient if the source server is on the same local network as the Identity Proxy.

To Configure All Indexes at Startup

The following example configures the entry-balancing request processor so that it primes the global index from the persisted file, if present, or from an external identity data stores source if necessary.

- Run the `dsconfig` tool to prime all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \  
  --processor-name dc_example_dc_com-eb-req-processor \  
  --set prime-all-indexes:true --set prime-index-source:file \  
  --set prime-index-source:ds
```

To Configure the Global Indexes Manually

If you do not want to configure priming during setup, you can configure index priming manually by creating an external server, creating a global attribute index, and then changing the entry-balancing request processor to load indexes from this external server.

1. Use the `dsconfig` tool to create an external server of the type UnboundID Identity Proxy to represent a peer of the Identity Proxy.

```
$ 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. 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 \  
  --set uid-index:global
```

```
--index-name uid \  
--set guaranteed-unique:true
```

3. Change the entry-balancing request processor to load the indexes at startup from the peer Identity Proxy 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 Identity Proxy accepts search result entries from the backend identity data stores.

To find the optimum rate, start low and specify a few thousand search entries per second. Then increase as necessary.

Working with Alternate Authorization Identities

Access control rules in an entry-balanced deployment are configured in the Identity Data Store backend servers and require access to the entry contents of the user *issuing* the request. This can introduce a possible issue when clients to the Identity Proxy authenticate as users whose entries are among the entry-balanced sets. If the server which is processing a request does not contain the issuing user's entry, then the access control cannot be evaluated.

For example, consider a deployment that has two entry-balancing sets, `set-01` and `set-02`. `Set-01` has entries in the range `uid=0-10000`, while `set-02` has entries for `uid=10001-20000`. The client with `uid=5000` binds to the Identity Proxy, which sends a `BIND` request to entry-balancing `set-01`. Next, the client sends a `SEARCH` request with filter `"(uid=15000)"`. The Identity Proxy determines that `uid=15000` lives on entry-balancing `set-02`. The Identity Proxy then determines that the entry for the authenticated user with `uid=5000` does not exist in `set-02` and that the access control handler would reject the `SEARCH` request issued by an unknown user.

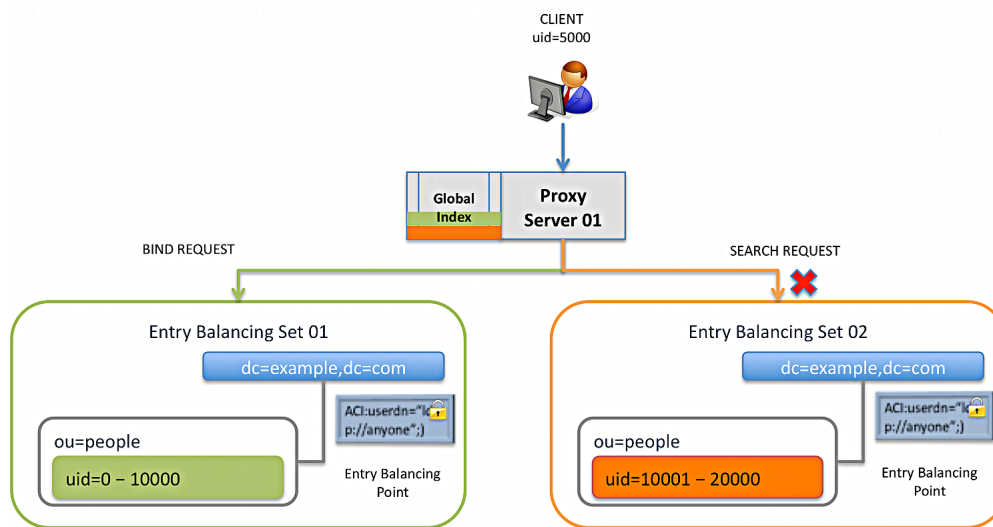


Figure 5: Entry-Balancing Issue with Clients Not Present in the Underlying Data Set

One solution to this problem is to make use of an *alternate authorization identity* for the user, which references an entry that exists in all Identity Data Stores in all backend sets and has an equivalent set of access control rights as the authenticated user. The alternate authorization identity is used when the Identity Proxy observes that the Identity Data Store processing a request does not contain the entry of the user issuing the request.

The following sections cover the procedures to configure the alternate authorization identities for the Identity Proxy.

About Alternate Authorization Identities

Whenever the Identity Proxy forwards a request to the backend set containing the user's entry, it forwards the request with an authorization identity that reflects the user's actual identity, since servers in that set already know about that user. However, when forwarding a request to a backend set that does *not* contain the user's entry, the Identity Proxy uses an *alternate authorization identity* that reflects the generic user with the same set of rights as the actual user issuing the request. Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries are not present within an entry-balanced dataset.

There are typically only a few different generic class of users from an access control perspective, which can be placed in a portion of the DIT that is not below the entry-balancing base DN and is replicated to all servers in the topology. For example, assume that you have three classes of users: full administrators, password administrators, and normal users. You could create the following entries in the topology and assign them the appropriate access rights:

- > uid=normal user,dc=example,dc=com
- > uid=server-admin,dc=example,dc=com
- > uid=password-admin,dc=example,dc=com

Returning to the example scenario, the client with uid=5000 binds to the Identity Proxy, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request for uid=15000. The Identity Proxy determines that uid=15000 lives on entry-balancing set-02. Next, the Identity Proxy then determines that the client uid=5000 does not have an entry on

entry-balancing set-02. The Identity Proxy uses an *alternate authorization identity* that reflects the generic user, `uid=normal user`, which has the same set of rights as the client `uid=5000` who is issuing the request. The access control is accepted and the SEARCH request returns a response for `uid=5000`.

Whenever a user authenticates to the Identity Proxy, the server can keep track of which backend set holds that user's entry and determine whether an alternate authorization identity is required. The server can also determine which of these generic accounts best describes the rights that the user should have.

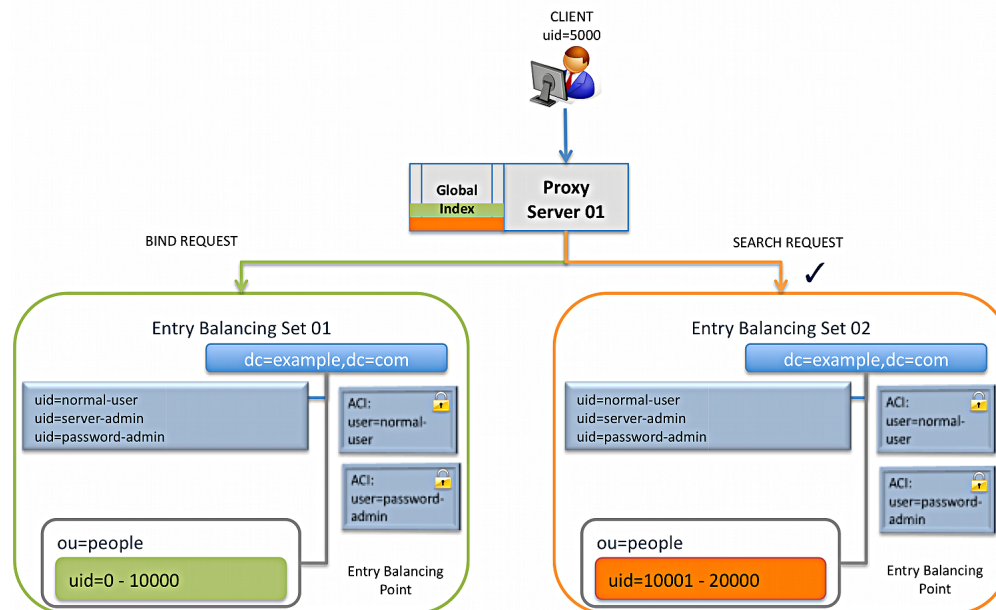


Figure 6: Alternate Authorization Identity Solves Access Control Issues in Entry-Balancing Deployments

When an alternate authorization identity is invoked, you will see `authzID='dn:uid=normal user,dc=example,dc=com'` in the server log, indicating that the alternate authorization identity was used. For example, if the user `.15000` is in a different backend set from user `.5000`, the log will show the following:

```
% bin/ldapsearch -D "uid=user.5000,ou=people,dc=example,dc=com" -w password \
  -b uid=user15000,ou=people,dc=example,dc=com "(objectclass=*)"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='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.15000,ou=people,dc=example,dc=com" scope=2
filter="(objectclass=*)" attrs="ALL"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2 resultCode=0
etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

Configuring Alternate Authorization Identities

Alternate authorization identities are specified by the `authz-attribute` property of the entry-balancing request processor configuration object. By default, the `authz-attribute` property has the default value of `ds-authz-map-to-dn`, which is an attribute reserved for this purpose.

To Configure Alternate Authorization Identity DNs

If a user entry has a value for `ds-authz-map-to-dn` whether it's explicitly contained in the entry or only present via a virtual attribute, then that will be used to specify the alternate authorization identity for the user. Otherwise, the default authorization identity (as indicated via the `authz-dn` configuration property) will be used to determine the alternate authorization identity.

1. Use `dsconfig` to set the `authz-dn` property of the entry-balancing request processor configuration. If any user among the balanced entries does not have an alternate authorization identity defined, the Identity Proxy will use the value of the `authz-dn` property of the entry-balancing request processor configuration.

```
$ bin/dsconfig set-request-processor-prop \  
  --processor-name dc_example_dc_com-eb-req-processor \  
  --set "authz-dn:uid=normal user,dc=example,dc=com"
```

2. Create an auxiliary object class containing `ds-authz-map-to-dn` as an allowed attribute.
3. Add the auxiliary object class value to all user entries of interest.
4. Then, add the following attribute value to a `server-admin` user.

```
ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
```

Chapter

7 Managing Entry-Balancing Replication

Replication in the UnboundID Identity Proxy 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*](#)
- [*Replication Prerequisites in an Entry-Balancing Deployment*](#)
- [*About the --restricted Argument of the dsreplication Command-Line Tool*](#)
- [*Checking the Status of Replication in an Entry-Balancing Deployment*](#)
- [*Example of Configuring Entry-Balancing Replication*](#)

Overview of Replication in an Entry-Balancing Environment

In an entry-balanced deployment, some data is replicated everywhere, such as the schema, the server registry, and other shared data, and some data is replicated only on certain servers. A replication domain contains all of the servers in a replicated topology and shares a schema. The replication domain is associated with the base 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 Identity Data Store is configured with a replication set name global configuration property, and two backends. One backend has a base DN that is replicated globally (such as `dc=example,dc=com`) and the second backend has a base DN associated with the entry-balancing point (such as `ou=people,dc=example,dc=com`).

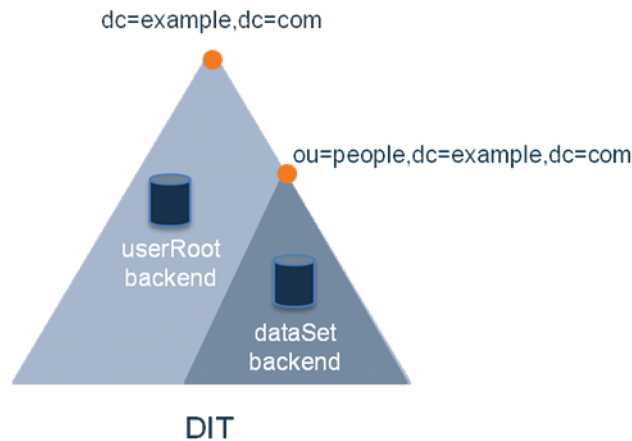


Figure 7: Global and Restricted Backends

If a data set name is not defined when you set up the Identity Proxy, one will be provided by default. The proper configuration of an entry-balancing environment requires coordination between the Identity Data Store and Identity Proxy. 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 Identity Data Store instances, you need two backends, a global backend for globally replicated data, such as `userRoot`, and a backend for the balancing point base DN, `dataSet`. Both backends need to be enabled for replication and initialized separately.

- **Replication set name.** Every Identity Data Store in your replicated topology must have a replication set name. This replication set name coordinates the Identity Proxy and the Identity Data Store. The restricted domain is only replicated within instances using the same replication set name.
- **Multiple Identity Proxy 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 `dsrepliation` 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 `dsrepliation` 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/dsrepliation status --hostname host1.example.com \  
--port 1389 --adminUID admin --adminPassword secret
```

```
--- Replication Status for dc=example,dc=com: Enabled ---  
  
Server      : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security  
-----:-----:-----:-----:-----:-----:-----  
  
austin1.example.com:1389 : 1000      : 0          : N/A          : 8989 : Enabled  
austin2.example.com:2389 : 1000      : 0          : N/A          : 8989 : Enabled  
newyork1.example.com:3389 : 1000      : 0          : N/A          : 8989 : Enabled  
newyork2.example.com:4389 : 1000      : 0          : N/A          : 8989 : Enabled  
  
  
-- Replication Status for ou=people,dc= example,dc=com (Set: set1): Enabled --  
  
Server      : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security  
-----:-----:-----:-----:-----:-----:-----  
  
austin1.example.com:1389 : 1000000    : 0          : N/A          : 8989 : Enabled  
austin2.example.com:2389 : 1000000    : 0          : N/A          : 8989 : Enabled  
  
  
---Replication Status for ou=people,dc= example,dc=com (Set: set2): Enabled ---  
  
Server      : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security  
-----:-----:-----:-----:-----:-----:-----  
  
newyork1.example.com:3389 : 1000000    : 0          : N/A          : 8989 : Enabled  
newyork2.example.com:4389 : 1000000    : 0          : N/A          : 8989 : Enabled
```

Example of Configuring Entry-Balancing Replication

This section describes how to set up a four-server replication topology that uses entry balancing to distribute entries across the servers. The procedure assumes that none of the servers have participated in any previous replication topology.

Assumptions

The example uses the LDAP (389) and replication (8989) ports respectively. It configures the following hosts:

- > austin1.example.com
- > newyork1.example.com
- > austin2.example.com
- > newyork2.example.com

In this example, we have a global domain of `dc=example,dc=com`, which is replicated across all servers. The data below the entry-balancing point of `ou=people,dc=example,dc=com` is distributed across two data sets, `dataSet1` and `dataSet2`. Each data set is replicated between two directory servers. Each of these servers is associated with one of two locations, Austin and New York.

Configuration Summary

To configure replication in an entry-balanced deployment, you must do the following:

- Install two identity data stores in an Austin location and two identity data stores in a New York location.
- Create a new backend, called `dataset`, to store the entry-balancing data set.
- Define entry-balancing set names `dataSet1` and `dataSet2` for assignment to the `replication-set-name` Global Configuration Property of the Identity Data Store instances.
- Import the data representing the global domain, stored in `userRoot`, into a server. Choose a server for each of the entry-balancing data sets, both stored in the backend named `dataset`.
- Enable replication and initialize remaining servers.
- Configure the identity proxies.
- Check the status of replication.

To Install the Identity Data Store

First, install the Identity Data Store instances. In this example, we install the following four servers, two in the Austin location and two in the New York location:

- > austin1.example.com
- > austin2.example.com
- > newyork1.example.com
- > newyork2.example.com

1. We install the first server, `austin1`, as follows:

```
root@austin1# ./setup --cli --baseDN dc=example,dc=com \
```

```
--ldapPort 389 --rootUserDN "cn=Directory Manager" \  
--rootUserPassword pass --no-prompt --acceptLicense
```

2. Install the second Austin server, `austin2`, in the same way:

```
root@austin2 # ./setup --cli --baseDN dc=example,dc=com \  
--ldapPort 389 --rootUserDN "cn=Directory Manager" \  
--rootUserPassword pass --no-prompt --acceptLicense
```

3. Next, install the two New York servers, `newyork1` and `newyork2`, as follows:

```
root@newyork1# ./setup --cli --baseDN dc=example,dc=com \  
--ldapPort 389 --rootUserDN "cn=Directory Manager" \  
--rootUserPassword pass --no-prompt --acceptLicense  
  
root@newyork# ./setup --cli --baseDN dc=example,dc=com \  
--ldapPort 389 --rootUserDN "cn=Directory Manager" \  
--rootUserPassword pass --no-prompt --acceptLicense
```

To Create the Database Backends and Define the Replication Set Name

1. On all servers, create the dataset backend as follows:

```
./bin/dsconfig --no-prompt create-backend \  
--backend-name dataset --type local-db --set enabled:true \  
--set base-dn:ou=people,dc=example,dc=com
```

2. Set the replication set name for `austin1.example.com` and `newyork1.example.com` to `dataset1`:

```
./bin/dsconfig --no-prompt \  
set-global-configuration-prop \  
--set replication-set-name:dataset1
```

3. Set the replication set name for `austin2.example.com` and `newyork1.example.com` to `dataset2`:

```
./bin/dsconfig --no-prompt \  
set-global-configuration-prop \  
--set replication-set-name:dataset2
```

To Create and Set the Locations

1. On the Austin servers, create the two locations, `newyork` and `austin`, and set the location of this instance to `austin`:

```
./bin/dsconfig --no-prompt create-location --location-name austin  
  
./bin/dsconfig --no-prompt create-location --location-name newyork \  
--set preferred-failover-location:austin  
  
./bin/dsconfig --no-prompt set-location-prop --location-name austin \  
--add preferred-failover-location:newyork  
  
./bin/dsconfig --no-prompt set-global-configuration-prop \  
--set location:austin
```

2. For the New York servers, set the location to `newyork`:

```
./bin/dsconfig --no-prompt create-location \  

```

```
--location-name austin

./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:newyork
```

To Import the Entries

We import the userRoot data, based on data defined in the userRoot.ldif file, into one server. This file does not contain entries at or within the entry-balancing point, ou=people,dc=example,dc=com.

1. Use the import-ldif command to import the userRoot data.

```
root@austin1# ./bin/import-ldif --backendID userRoot \
--ldifFile /data/userRoot.ldif \
--includeBranch dc=example,dc=com \
--rejectFile /data/austin1-import-rejects \
--port 389
--hostname austin1.example.com
```

2. Import the dataSet1 data on one server into the dataset backend, which is assigned the dataSet1 replication-set-name.

```
root@austin1# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset1.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin1-dataset-import-rejects \
--hostname austin1.example.com --port 389
```

3. Import the dataSet2 data on one server into the dataset backend, which is assigned the dataSet2 replication-set-name.

```
root@austin2# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset2.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin2-dataset-import-rejects \
--hostname austin2.example.com --port 389
```

To Enable Replication in an Entry-Balancing Deployment

Now we can enable replication between the servers and initialize the remaining servers without data. Notice that we specify the --restricted domain in the dsreplication command.

1. Run dsreplication enable to enable the servers in the topology. The first invocation of this command creates the admin account.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 austin2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
```

```
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

2. Enable replication between austin1 and newyork1. This procedure automatically enables replication between austin2 and newyork1 as well.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork1.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

3. Enable replication between austin1 and newyork2. This will complete the entry-balancing replication setup.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

4. Initialize the remaining servers without data. The global domain, dc=example,dc=com needs to be initialized on austin2, newyork1 and newyork2. The ou=people,dc=example,dc=com entry-balancing domain needs to be initialized from austin1 to newyork2, and then again from austin2 to newyork2. We will combine these steps by initializing both domains with one invocation once austin2 is initialized with the global domain.

```
root@austin1# ./bin/dsreplication initialize --hostSource
austin1.example.com --portSource 389 \
--hostDestination austin2.example.com --portDestination 389
--adminUID admin \
--adminPassword password --baseDN dc=example,dc=com --no-
prompt

root@austin1# ./bin/dsreplication initialize --hostSource
austin1.example.com --portSource 389 \
--hostDestination newyork1.example.com --portDestination 389
--adminUID admin \
--adminPassword password --baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com --no-prompt

root@austin2# ./bin/dsreplication initialize --hostSource
austin2.example.com --portSource 389 \
--hostDestination newyork2.example.com --portDestination 389
--adminUID admin \
--adminPassword password --baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com --no-prompt
```

To Check the Status of Replication

Once replication has been configured, check the status of the replication topology using the `dsreplication status` command.

- Run the `dsreplication status` command to check its status.

```
root@austin1# ./bin/dsreplication status \  
--adminPassword pass --no-prompt --port 389
```


Chapter

8

Managing Security

The UnboundID Identity Proxy provides a full suite of security features to secure communication between the client and the server, to establish trust between components (for example, for replication and administration), and to secure data. Internally, the Identity Proxy uses cryptographic mechanisms that leverage the Java JRE's Java Secure Sockets Extension (JSSE) implementation of the SSL protocol using Key Manager and Trust Manager providers for secure connection integrity and confidentiality, and the Java Cryptography Architecture (JCA) for data encryption.

This chapter presents procedures to configure security and covers the following topics:

Topics:

- [*Summary of the UnboundID Identity Proxy Security Features*](#)
- [*Identity Proxy SSL and StartTLS Support*](#)
- [*Managing Certificates*](#)
- [*Configuring the Key and Trust Manager Providers*](#)
- [*Configuring SSL in the Identity Proxy*](#)
- [*Configuring StartTLS*](#)
- [*Authentication Mechanisms*](#)
- [*Working with SASL Authentication*](#)
- [*Configuring Pass-Through Authentication*](#)
- [*Adding Operational Attributes that Restrict Authentication*](#)
- [*Configuring Certificate Mappers*](#)

Summary of the UnboundID Identity Proxy Security Features

The UnboundID Identity Proxy supports a strong set of cryptographic and other mechanisms to secure communication and data. The following security-related features are available:

- **SSL/StartTLS Support.** The Identity Proxy supports the use of SSL and StartTLS to encrypt communication between the client and the server. Administrators can configure different certificates for each connection handler, or use the same certificate for all connection handlers. Additionally, the server allows for more fine-grained control of the key material used in connecting peers in SSL handshakes and trust material for storing certificates.
- **Message Digest/Encryption Algorithms.** The Identity Proxy supports the use of a number of one-way message digests (e.g., CRYPT, 128-bit MD5, 160-bit SHA-1, and 256-bit, 384-bit, and 512-bit SHA-2 digests with or without salt) as well as a number of reversible encryption algorithms (BASE64, 3DES, AES, RC4, and Blowfish) for storing passwords. Note that even if passwords are encoded using reversible encryption, that encryption is intended for use only within the server itself, and the passwords will not be made available to administrators in unencrypted form. It is generally recommended that encrypted password storage only be used if you anticipate using an authentication mechanism that requires the server to have access to the clear-text representation of passwords, like CRAM-MD5 or DIGEST-MD5.
- **SASL Mechanism Support.** The Identity Proxy supports a number of SASL mechanisms, including ANONYMOUS, CRAM-MD5, DIGEST-MD5, EXTERNAL, PLAIN, and GSSAPI. In some vendors' directory servers, the use of CRAM-MD5 and DIGEST-MD5 requires that the server have access to the clear-text password for a user. In this case, the server supports reversible encryption to store the passwords in a more secure encoding than clear text. The server also supports two types of one-time password (OTP) mechanisms for multi-factor authentication: UNBOUNDID-TOTP SASL and UNBOUNDID-DELIVERED-OTP SASL. The proprietary UNBOUNDID-TOTP SASL mechanism allows multi-factor authentication to the server using the time-based one-time password (TOTP) code. The proprietary UNBOUNDID-DELIVERED-OTP SASL mechanism allows multi-factor authentication to the server by delivering a one-time password to the the end user through some out-of-band channel, such as email or SMS.
- **Password Policy Support.** The Identity Proxy provides extensive password policy support including features, like customizable password attributes, maximum password age, maximum password reset age, multiple default password storage schemes, account expiration, idle account lockout and others. The server also supports a number of password storage schemes, like one-way digests (CRYPT, MD5, SMD5, SHA, SSHA, SSHA256, SSHA384, SSHA512) and reversible encryption (BASE64, 3DES, AES, RC4, BLOWFISH). Administrators can also use a number of password validators, like maximum password length, similarity to current password and the set of characters used. See the chapter on *Password Policies* for more information.
- **Full-Featured Access Control System.** The Identity Proxy provides a full-featured access control subsystem that determines whether a given operation is allowable based on a wide

range of criteria. The access control system allows administrators to grant or restrict access to data, restrict the use of specific types of controls and extended operations and provides strong validation for access control rules before accepting them. See the chapter on *Access Control* for more information.

- **Client Connection Policies Support.** The Identity Proxy provides the ability to control which clients get connected to the server, how they can get connected to the system, and what resources or operations are available to them. For example, administrators can set up client connection criteria that blacklists IP addresses or domains that are known to attempt brute force attacks. Likewise, client connection policies can be configured to restrict the type of operations, controls, extended-operations, SASL mechanisms, search filters and resource limits available to the client. For example, you can configure a client connection policy that limits the number of concurrent connections or rejects all requests on unsecured connections.
- **Backup Protection.** The Identity Proxy provides the ability to protect the integrity of backup contents using cryptographic digests and encryption. When generating a backup, the administrator has an option to generate a cryptographic digest of the backup contents and also optionally to digitally sign that digest. The server also has options to compress and/or encrypt the contents of the backup. When restoring the backup, the server can verify that the digest matches the content of the backup and generates an error if the backup has been changed from when it was initially written, making it tamper-evident. The server also provides the ability to verify the integrity of a backup without actually restoring it. See chapter on *Backing Up and Restoring Data* for more information.

Identity Proxy SSL and StartTLS Support

The UnboundID Identity Proxy supports the use of SSL and/or StartTLS to secure communication with clients and other components in your environment.



Note: Although the term "SSL" (Secure Sockets Layer) has been superseded by "TLS" (Transport-Layer Security), the older term "SSL" will continue to be used in this document to make it easier to distinguish between the use of TLS as a general mechanism for securing communication and the specific use of the StartTLS extended operation.

The supported versions of SSL or StartTLS are determined by what the underlying JVM supports. The server will automatically look at the supported protocols and attempt to determine the best one to use.

When using Oracle Java SE 1.7, version TLSv1.2 is preferred by the server. A particular protocol can be specified by setting the `com.unboundid.util.SSLUtil.defaultSSLProtocol` property

LDAP-over-SSL (LDAPS)

The Identity Proxy provides the option of using dedicated connection handlers for LDAPS connections. LDAPS differs from LDAP in that upon connect, the client and server establish an SSL session before any LDAP messages are transferred. LDAPS connection handlers with SSL enabled may only be used for secure communication, and connections must be closed when the SSL session is shut down.

StartTLS Support

The StartTLS extended operation provides a means to add SSL encryption to an existing plain-text LDAP connection. The client opens an unencrypted TCP connection to the server and, after processing zero or more LDAP operations over that clear-text connection, sends a StartTLS extended request to the server to indicate that the client-server communication should be encrypted.

To require the use of SSL for client connections accepted by a connection handler, set `use-ssl` to true for that connection handler. To allow clients to use StartTLS on a connection handler, the administrator must configure that connection handler to allow StartTLS. Because SSL and StartTLS are mutually exclusive, you cannot enable both SSL and StartTLS for the same connection handler (although you can have some connection handlers configured to use SSL and others configured to use StartTLS).

Managing Certificates

You can generate and manage certificates using a variety of commonly available tools, such as the Java `keytool` utility, which is a key and certificate management utility provided with the Java SDK. The `keytool` utility can be used to create keystores, which hold key material used in the course of establishing an SSL session, and truststores, which may be consulted to determine whether a presented certificate should be trusted.

Because there are numerous ways to create or obtain certificates, the procedures in this section will only present basic steps to set up your certificates. Many companies have their own certificate authorities or have existing certificates that they use in the servers, and in such environments you should follow the guidelines specific to your company's implementation.

The UnboundID Identity Proxy supports three keystore types: Java Keystore (JKS), PKCS#12, and PKCS#11.

- **Java Keystore (JKS).** In most Java SE implementations, the JKS keystore is the default and preferred keystore format. JKS keystores may be used to hold certificates for other Java-based applications, but such keystores are likely not compatible with non-Java-based applications.
- **PKCS#12.** This keystore type is a well-defined standard format for storing a certificate or certificate chain, and may be used to hold certificates already in use for other types of

servers. Most other servers that provide a proprietary format for storing certificates provide a mechanism for converting those certificates to PKCS#12.

- **PKCS#11.** Also, known as Cryptoki (pronounced "crypto-key") is a format for cryptographic token interfaces for devices, such as cryptographic smart cards, hardware accelerators, and high performance software libraries. PKCS#11 tokens may also offer a higher level of security than other types of keystores, and many of them have been FIPS 140-2 certified and may be tamper-evident or tamper-resistant.

Authentication Using Certificates

The Identity Proxy supports two different mechanisms for certificate-based authentication:

- **Client Certificate Validation.** The Identity Proxy can request the client to present its own certificate for client authentication during the SSL or StartTLS negotiation process. If the client presents a certificate, then the server will use the trust manager provider configured for the associated connection handler to determine whether to continue the process of establishing the SSL or StartTLS session. If the client certificate is not accepted by the trust manager provider, then the server will terminate the connection. Note that even if the client provides its own certificate to the server during the process of establishing an SSL or StartTLS session, the underlying LDAP connection may remain unauthenticated until the client sends an LDAP bind request over that connection.
- **SASL EXTERNAL Certificate Authentication.** The SASL EXTERNAL mechanism is used to allow a client to authenticate itself to the Identity Proxy using information provided outside of LDAP communication. In the Identity Proxy, that information must come in the form of a client certificate presented during the course of SSL or StartTLS negotiation. Once the client has established a secure connection to the server in which it provided its own client certificate, it may send a SASL EXTERNAL bind request to the server to request that the server attempt to identify the client based on information contained in that certificate. The server will then use a certificate mapper to identify exactly one user entry that corresponds to the provided client certificate, and it may optionally perform additional verification (e.g., requiring the certificate the client presented to be present in the `userCertificate` attribute of the user's entry). If the certificate mapper cannot identify exactly one user entry for that certificate, or if its additional validation is not satisfied, then the bind attempt will fail and the client connection will remain unauthenticated.

Creating Server Certificates using Keytool

Generate and manage certificates using the `keytool` utility, which is available in the Java SDK. The `keytool` utility is a key and certificate management utility that allows users to manage their public/private key pairs, x509 certificate chains and trusted certificates. The utility also stores the keys and certificates in a keystore, which is a password-protected file with a default format of JKS although other formats like PKCS#12 are available. Each key and trusted certificate in the keystore is accessed by its unique alias.

To Create a Server Certificate using Keytool

1. Change to the directory where the certificates will be stored.

```
$ cd /ds/UnboundID-Proxy/config
```

2. Use the `keytool` utility to create a private/public key pair and a keystore. The `keytool` utility is part of the Java SDK. If you cannot access the utility, make sure to change your path to include the Java SDK (`${JAVA_HOME}/bin`) directory.

The following command creates a keystore named "keystore", generates a public/private key pair and creates a self-signed certificate based on the key pair. This certificate can be used as the server certificate or it can be replaced by a CA-signed certificate chain if additional `keytool` commands are executed. Self-signed certificates are only convenient for testing purposes, they are not recommended for use in deployments in which the set of clients is not well-defined and carefully controlled. If clients are configured to blindly trust any server certificate, then they may be vulnerable to man-in-the-middle attacks.

The `-dname` option is used to specify the certificate's subject, which generally includes a CN attribute with a value equal to the fully-qualified name that clients will use to communicate with the Identity Proxy. Some clients may refuse to establish an SSL or StartTLS session with the server if the certificate subject contains a CN value which does not match the address that the client is trying to use, so this should be chosen carefully. If the `-dname` option is omitted, you will be prompted for input. The certificate will be valid for 180 days.

```
$ keytool -genkeypair \
-dname "CN=server.example.com,ou=Identity Proxy Certificate,
O=Example Company,C=US" \
-alias server-cert \
-keyalg rsa \
-keystore keystore \
-keypass changeit \
-storepass changeit \
-storetype JKS \
-validity 180 \
-noprompt
```



Note: The `-keypass` and `-storepass` arguments can be omitted to cause the tool to interactively prompt for the password. Also, the key password should match the keystore password.

3. View the keystore. Notice the entry type is `privateKeyEntry` which indicates that the entry has a private key associated with it, which is stored in a protected format to prevent unauthorized access. Also note that the Owner and the Issuer are the same, indicating that this certificate is self-signed.

```
$ keytool -list -v -keystore keystore -storepass changeit
```

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: server-cert
Creation date: Sep 30, 2011
Entry type: PrivateKeyEntry
```

```
Certificate chain length: 1
Certificate[1]:
Owner: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company, C=US
Issuer: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company,
C=US
Serial number: 4ac3695f
Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29 09:21:19 CDT 2012
Certificate fingerprints:
MD5: 3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00
SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
Signature algorithm name: SHA1withRSA
Version: 3
```

4. If you are going to have your certificate signed by a Certificate Authority, skip to step 7. Otherwise export the self-signed certificate. Then examine the certificate.

```
$ keytool -export -alias server-cert -keystore keystore -rfc -file server.crt
Enter keystore password:
Certificate stored in file <server.crt>
```

```
$ cat server.crt
-----BEGIN CERTIFICATE-----
MIICVTCCAb6gAwIBAgIESsNpXzANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEwJVUzEYMBYGA1UEChMPRXhhb
XBsZSBDb2lwYW55MS8wLQYDVQQLEyZVbmJvdW5kaWQgRGlyZW50b3J5IFNlcnZlciBD ZXJ0aWZpY2F0ZTEV
BMGA1UEAxMMMTcyLjE2LjE5My4xMB4XDTA5MDkzMDE0MjExOVoXDTEwMDMyOTE0MjExOVoWb3ZELMakGA1UEBh
MCVVMxGDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEvMC0GA1UE CxMmVW5ib3VuZG1kIERpcmVjdG9yeSBZRX
J2ZXIgc2Vydg1maWNhdGUxFTATBgNVBAMTDDE3Mi4x Ni4xOTMuMTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwG
YkCgYEAuRBPSeRcqr4XP8PjJWcGDVR3lwE cItmMImbjpf0rTq+KG8Ssp8+se+LjLHLaeNg3itR3xMBwp7mQ4
E42i2PBIIZ0PwOKBRPzXDxpsITsSy3o9anTsopIVglUpST2iHGBQ+j+VY33cdcc5EoJwYykZ4dliu45yc834
VByXjiKUCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCJIZfsfQuUig4F0kPC/0fFbhW96TrLTOi6AMIOtork1SuJ
lkxp/nT+eD8eGoE+zshyJWTFvVnzMDIIFMjwDIIVvnyYmyerlVlCchyJE6JyFiLpBWs6RuLD8iuHydYEWK8NkEF
YvVb/UIKqJlZ8H8+1lppt0bENRnGD7zMWJv5ZE49w==
-----END CERTIFICATE-----
```

5. Import the self-signed certificate into a truststore, and then type `yes` to trust the certificate.

```
$ keytool -importcert -alias server-cert -file server.crt \
-keystore truststore -storepass changeit
```

```
Owner: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company, C=US
Issuer: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company,
C=US
Serial number: 4ac3695f
Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29 09:21:19 CDT 2012
Certificate fingerprints:
MD5: 3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00
SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
Signature algorithm name: SHA1withRSA
Version: 3
Trust this certificate? [no]: yes
Certificate was added to keystore
```

6. View the truststore with the self-signed certificate. If you intend to use this self-signed certificate as your server certificate, you are done. *Again, it is not recommended to use self-signed certificate in production deployments.* Note that the entry type of `trustedCertEntry` indicates that the keystore owner trusts that the public key in the certificate belongs to the entity identified by the owner of the certificate.

```
$ keytool -list -v -keystore truststore -storepass changeit
```

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: server-cert
Creation date: Sep 30, 2011
Entry type: trustedCertEntryOwner: CN=server.example.com, OU=Identity Proxy
Certificate, O=Example Company, C=US
Issuer: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company,
C=US
```

```
Serial number: 4ac3695f Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29
09:21:19 CDT 2012
Certificate fingerprints:
MD5: 3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00
SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
Signature algorithm name: SHA1withRSA
Version: 3
```

7. Create the Certificate Signing Request (CSR) by writing to the file `server.csr`. Follow the instructions of the third-party Certificate Authority (CA), and submit the file to a CA. The CA authenticates you and then returns a certificate reply, which you can save as `signed.crt`.

```
$ keytool -certreq -v -alias server-cert -keystore keystore \
-storepass changeit -file server.csr
```

```
Certification request stored in file <server.csr>
Submit this to your CA
```

8. If you are working with a third-party CA or if your company has your own CA server, then both the key and trust stores should include information about the CA's root certificate as well as any intermediate certificates used to sign the Identity Proxy certificate. Obtain the CA root and any intermediate certificates to set up a chain of trust in your keystore. View the trusted CA and intermediate certificates to check that the displayed certificate fingerprints match the expected ones.

```
$ keytool -v -printcert -file root.crt
$ keytool -v -printcert -file intermediate.crt
```

9. Import the CA's root certificate in the keystore and truststore. If there are other intermediate certificates, then import them using the same commands, giving them each different aliases in the key and trust stores.

```
$ keytool -importcert -v -trustcacerts -alias cacert \
-keystore keystore -storepass changeit -file root.crt
$ keytool -importcert -v -trustcacerts -alias cacert -keystore truststore \
-storepass changeit -file root.crt
```

10. Import the Identity Proxy certificate signed by the CA into your keystore, which will replace the existing self-signed certificate created when the private key was first generated.

```
$ keytool -importcert -v -trustcacerts -alias server-cert -keystore keystore -
storepass changeit -file signed.crt
```

```
Owner: CN=server.example.com, OU=Identity Proxy Certificate, O=Example Company, C=US
Issuer: EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate Authority,
O=Example Company, L=Austin, ST=Texas, C=US
Serial number: e19cb2838441dbb6 Valid from: Wed Sep 30 10:10:30 CDT 2011 until: Thu
Sep 30 10:10:30 CDT 2012
Certificate fingerprints:
MD5: E0:C5:F7:CF:0D:13:F5:FC:2D:A6:A4:87:FD:4C:36:1A
SHA1: E4:15:0B:ED:99:1C:13:47:29:66:76:A0:3B:E3:4D:60:33:F1:F8:21
Signature algorithm name: SHA1withRSA
Version: 1
Trust this certificate? [no]: yes
Certificate was added to keystore [Storing changeit]
```

11. Add the certificate to the truststore.

```
$ keytool -importcert -v -trustcacerts -alias server-cert \
-keystore truststore -storepass changeit -file signed.crt
```

Client Certificates

Client certificates can be used when stronger client authentication is desired but are not required for SSL connections to be established. The process for creating client certificates usually involve following an organization's certificate management policies. There are two important considerations to take into account:

- If a client presents its own certificate to the server, then the server must trust that certificate. This generally means that self-signed client certificates are not acceptable for anything but testing purposes or cases in which there are very small number of clients that will be presenting their own certificates. Otherwise, it is not feasible to configure the server to trust every client certificate.
- If the client certificates will be used for LDAP authentication via SASL EXTERNAL, then the certificate must contain enough information to allow the Identity Proxy to associate it with exactly one user entry. The requirements for this are dependent upon the certificate mapper configured for use in the server, but this may impose constraints on the certificate (for example, the format of the certificate's subject).

Creating PKCS#12 Certificates

PKCS#12 is an industry standard format for deploying X.509 certificates (or certificate chains) and a private key as a single file. PKCS#12 is part of the family of standards called the Public-Key Cryptography Standard (PKCS) developed by RSA Laboratories.

To Generate PKCS#12 Certificates using Keytool

- To create a new certificate in PKCS#12 format, follow the same procedures as in [Creating Server Certificates using Keytool](#), except use the `--storetype pkcs12` argument. For example, to create a PKCS#12 self-signed certificate and keystore, use the following command:

```
$ keytool -genkeypair \
  -dname "CN=server.example.com,ou=Identity Proxy Certificate,O=Example Company,C=US" \
  -alias server-cert -keyalg rsa -keystore keystore.pk12 -keypass changeit \
  -storepass changeit -storetype pkcs12 -validity 180 -noprompt
```

To Export a Certificate from an NSS Database in PKCS#12 Format

Some directory servers, including the Sun/Oracle DSEE Directory Server, use the Network Security Services (NSS) library to manage certificates. If you have such a directory server and wish to migrate its certificates for use with the UnboundID Identity Proxy, then PKCS#12 can be used to accomplish this task. Use the `pk12util` NSS command-line utility to export a certificate from an NSS certificate database in PKCS12 format. You can use the PKCS#12 certificate when using QuickSetup or setting up SSL.

- Run the following `pk12util` command.

```
$ pk12util -o server.p12 -n server-cert -k /tmp/pwdfile \  
-w /tmp/pwdfile -d . -P "ds-"
```

```
nss-pk12util: PKCS12 EXPORT SUCCESSFUL
```

Working with PKCS#11 Tokens

The Cryptographic Token Interface Standard, PKCS#11, defines the native programming interfaces to cryptographic tokens, like Smartcards and hardware cryptographic accelerators. A security token provides cryptographic services. PKCS#11 provides an interface to cryptographic devices via "slots". Each slot, which corresponds to a physical reader or other device interface, may contain a token. A token is typically a PKCS#11 hardware token implemented in physical devices, such as hardware accelerators or smart cards. A software token is a PKCS#11 token implemented entirely in software.



Note: Because different types of PKCS#11 tokens have different mechanisms for creating, importing, and managing certificates, it may or may not be possible to achieve this using common utilities like `keytool`. In some cases (particularly for devices with strict Note FIPS 140-2 compliance), it may be necessary to use custom tools specific to that PKCS#11 token for managing its certificates. Consult the documentation for your PKCS#11 token for information about how to configure certificates for use with that token.

Configuring the Key and Trust Manager Providers

Java uses key managers to get access to certificates to use for SSL and StartTLS communication. Administrators use the Identity Proxy's key manager providers to provide access to keystore contents. There are three types of key manager providers:

- **JKS Key Manager Provider.** Provides access to certificates stored in keystores using the Java-default JKS format.
- **PKCS#11 Key Manager Provider.** Provides access to certificates maintained in PKCS#11 tokens.
- **PKCS#12 Key Manager Provider.** Provides access to certificates in PKCS#12 files.

Trust manager providers are used to determine whether to trust any client certificate that may be presented during the process of SSL or StartTLS negotiation. The available trust manager provider types include:

- **Blind Trust Manager Provider.** Automatically trusts any client certificate presented to the server. This should only be used for testing purposes. Never use it for production environments, because it can be used to allow users to generate their own certificates to impersonate other users in the server.

- **JKS Trust Manager Provider.** Attempts to determine whether to trust a client certificate, or the certificate of any of its issuers, is contained in a JKS-formatted file.
- **PKCS#12 Trust Manager Provider.** Attempts to determine whether to trust a client certificate, or the certificate of any of its issuers, is contained in a PKCS#12 file.

Configuring the JKS Key and Trust Manager Provider

The following procedures are identical to those in the previous section except that the `dsconfig` tool in non-interactive mode commands are presented from the command line.

To Configure the JKS Key Manager Provider

1. Change to the server root.

```
$ cd /ds/UnboundID-Proxy
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Identity Proxy user.

```
$ echo 'changeit' > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the `dsconfig` tool to enable the key manager provider.

```
$ bin/dsconfig set-key-manager-provider-prop \
  --provider-name JKS --set enabled:true
```

4. Use `dsconfig` to enable the trust manager provider.

```
$ bin/dsconfig set-trust-manager-provider-prop \
  --provider-name JKS --set enabled:true
```

5. Use `dsconfig` to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the `--set ssl-cert-nickname:<aliasname>` to set it, or you can let the server decide by using the `--reset ssl-cert-nickname` option. For example, if the server certificate has an alias of "server," add the option `--set ssl-cert-nickname:server` to the command.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set listen-port:1636 --set enabled:true
```

6. Test the listener port for SSL-based client connection on port 1636 to return the Root DSE. Type `yes` to trust the certificate.

```
$ bin/ldapsearch --port 1636 --useSSL --baseDN "" --searchScope base \
  "(objectclass=*)"
```

```
The server is using the following certificate:
  Subject DN: CN=179.13.201.1, OU=Identity Proxy Certificate, O=Example Company,
  L=Austin, ST=Texas, C=US
```

```
Issuer DN: EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate
Authority, O=Example Company, L=Austin, ST=Texas, C=US
Validity: Fri Sep 25 15:21:10 CDT 2011 through Sat Sep 25 15:21:10 CDT 2012
Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes
```

7. If desired, you may disable the LDAP Connection Handler so that communication can only go through SSL.

```
$ bin/dsconfig set-connection-handler-prop \
--handler-name "LDAP Connection Handler" \
--set enabled:false
```

Configuring the PKCS#12 Key Manager Provider

PKCS#12 (sometimes referred to as the Personal Information Exchange Syntax Standard) is a standard file format used to store private keys with its accompanying public key certificates, protected with a password-based symmetric key.

To Configure the PKCS#12 Key Manager Provider

1. Change to the identity data store root.

```
$ cd /ds/UnboundID-Proxy
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Identity Proxy user.

```
$ echo 'changeit' > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the dsconfig tool to configure and enable the PKCS#12 key manager provider.

```
$ bin/dsconfig set-key-manager-provider-prop \
--provider-name PKCS12 \
--set enabled:true \
--set key-store-file:/config/keystore.p12 \
--set key-store-type:PKCS12 \
--set key-store-pin-file:/config/keystore.pin
```

4. Use the dsconfig tool to configure and enable the PKCS#12 trust manager provider.

```
$ bin/dsconfig set-trust-manager-provider-prop \
--provider-name PKCS12 \
--set enabled:true \
--set trust-store-file:/config/truststore.p12
```

5. Use dsconfig to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the --set ssl-cert-nickname:<aliasname> to set it, or you can let the server decide by using the --reset ssl-cert-nickname option. For example, if the server certificate has an alias of "server," add the option --set ssl-cert-nickname:server to the command.

```
$ bin/dsconfig set-connection-handler-prop \
```

```
--handler-name "LDAPS Connection Handler" \
--set enabled:true \
--set listen-port:2636 \
--set ssl-cert-nickname:1 \
--set key-manager-provider:PKCS12 \
--set trust-manager-provider:PKCS12
```

Configuring the PKCS#11 Key Manager Provider

The Cryptographic Token Interface (Cryptoki), or PKCS#11, format defines a generic interface for cryptographic tokens used in single sign-on or smartcard systems. The Identity Proxy supports PKCS#11 keystores.

To Configure the PKCS#11 Key Manager Provider

1. Change to the server root.

```
$ cd /ds/UnboundID-Proxy
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Identity Proxy user.

```
$ echo 'changeit' > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the `dsconfig` tool to configure and enable the PKCS#11 key manager provider.

```
$ bin/dsconfig set-key-manager-provider-prop \
--provider-name PKCS11 \
--set enabled:true \
--set key-store-type:PKCS11 \
--set key-store-pin-file:/config/keystore.pin
```

4. Use the `dsconfig` tool to enable the trust manager provider. Since there is no PKCS#11 trust manager provider, then you must use one of the other truststore provider types (for example, JKS or PKCS#12).

```
$ bin/dsconfig set-trust-manager-provider-prop \
--provider-name JKS \
--set enabled:true \
--set trust-store-file:/config/truststore.jks
```

5. Use `dsconfig` to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the `--set ssl-cert-nickname:<aliasname>` to set it, or you can let the server decide by using the `--reset ssl-cert-nickname` option. For example, if the server certificate has an alias of "server," add the option `--set ssl-cert-nickname:server` to the command.

```
$ bin/dsconfig set-connection-handler-prop \
--handler-name "LDAPS Connection Handler" \
--set enabled:true \
--set listen-port:1636 \
--set ssl-cert-nickname:1 \
--set key-manager-provider:PKCS11 \
--set trust-manager-provider:JKS
```

Configuring the Blind Trust Manager Provider

The Blind Trust Manager provider accepts any peer certificate presented to it and is provided for testing purposes only. This trust manager should not be used in production environments, because it can allow any client to generate a certificate that could be used to impersonate any user in the server.

To Configure the Blind Trust Manager Provider

1. Change to the Identity Proxy install root.

```
$ cd /ds/UnboundID-Proxy
```

2. Use the `dsconfig` tool to enable the blind trust manager provider.

```
$ bin/dsconfig set-trust-manager-provider-prop \  
--provider-name "Blind Trust" --set enabled:true
```

3. Use `dsconfig` to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the `--set ssl-cert-nickname:<aliasname>` to set it, or you can let the server decide by using the `--reset ssl-cert-nickname` option. For example, if the server certificate has an alias of "server," add the option `--set ssl-cert-nickname:server` to the command.

Configuring SSL in the Identity Proxy

The UnboundID Identity Proxy provides a means to enable SSL or StartTLS at installation time, using either an existing certificate or by automatically generating a self-signed certificate. However, if SSL was not configured at install time, then it may be enabled at any time using the following process. These instructions assume that the certificate is available in a JKS-formatted keystore, but a similar process may be used for certificates available through other mechanisms like a PKCS#12 file or a PKCS#11 token.

To Configure SSL in the Identity Proxy

1. Change to the server root directory.

```
$ cd /ds/UnboundID-Proxy
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Identity Proxy user.

```
$ echo 'changeit' > config/keystore.pin  
$ chmod 0400 config/keystore.pin
```

3. Run the `dsconfig` command with no arguments in order to launch the `dsconfig` tool in interactive mode. Enter the connection parameters when prompted.
4. On the **Identity Proxy Configuration Console main** menu, enter `o` (lowercase letter "o") to change the complexity of the configuration objects menu. Select the option to show objects at the Standard menu.
5. On the **Identity Proxy Configuration Console main** menu, enter the number corresponding to the Key Manager Provider.
6. On the **Key Manager Provider management** menu, select the option to view and edit an existing key manager.
7. On the **Key Manager Provider** menu, enter the option for JKS. You will see other options, like Null, PKCS11, and PKCS12.
8. Make any necessary changes to the JKS key manager provider for the keystore that you will be using. The `enabled` property must have a value of `TRUE`, the `key-store-file` property must reflect the path to the keystore file containing the server certificate, and the `key-store-pin-file` property should reflect the path to a file containing the password to use to access the keystore contents.
9. On the **Enabled Property** menu, enter the option to change the value to `TRUE`.
10. On the **File Based Key Manager Provider**, type `f` to save and apply the changes.
11. Return to the **dsconfig main** menu, and enter the number corresponding to Trust Manager Provider.
12. On the **Trust Manager Provider management** menu, enter the option to view and edit an existing trust manager provider.
13. On the **Trust Manager Provider** menu, enter the option for JKS. You will see other options for Blind Trust (accepts any certificate) and PKCS12 reads information about trusted certificates from a PKCS#12 file.
14. Ensure that the JKS trust manager provider is enabled and that the `trust-store-file` property has a value that reflects the path to the truststore file to consult when deciding whether to trust any presented certificates.
15. On the **File Based Trust Manager Provider** menu, type `f` to save and apply the changes.
16. Return to the **dsconfig main** menu, enter the number corresponding to Connection Handler.
17. On the **Connection Handler management** menu, enter the option to view and edit and existing connection handler.
18. On the **Connection Handler** menu, enter the option for LDAPS Connection Handler. You will see other options for JMX Connection Handler and LDAP Connection Handler.
19. On the **LDAP Connection Handler** menu, ensure that the connection handler has an appropriate configuration for use. The `enabled` property should have a value of `TRUE`, the

listen-port property should reflect the port on which to listen for SSL-based connections, and the ssl-cert-nickname property should reflect the alias for the target certificate in the selected keystore. Finally, when completing the changes, type `f` to save and apply the changes.

20. Verify that the server is properly configured to accept SSL-based client connections using an LDAP-based tool like `ldapsearch`. For example:

```
$ bin/ldapsearch --port 1636 --useSSL --baseDN "" \
--searchScope base "(objectclass=*)"
```

```
The server is using the following certificate:
Subject DN: CN=179.13.201.1, OU=Identity Proxy
Certificate, O=Example Company, L=Austin, ST=Texas,
C=US Issuer DN: EMAILADDRESS=whatever@example.com,
CN=Cert Auth, OU=My Certificate Authority, O=Example
Company, L=Austin, ST=Texas, C=US
Validity: Fri Sep 25 15:21:10 CDT 2011 through Sat Sep 25 15:21:10 CDT 2012
Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes
```

21. If desired, you may disable the LDAP connection handler so only the LDAPS connection handler will be enabled and the server will only accept SSL-based connections.

Configuring StartTLS

The StartTLS extended operation is used to initiate a TLS-secured communication channel over a clear-text connection, such as an insecure LDAP connection. The main advantage of StartTLS is that it provides a way to use a single connection handler capable of both secure and insecure communication rather than requiring a dedicated connection handler for secure communication.

To Configure StartTLS

1. Use `dsconfig` to configure the Connection Handler to allow StartTLS. The `allow-start-tls` property cannot be set if SSL is enabled. The connection handler must also be configured with a key manager provider and a trust manager provider.

```
$ bin/dsconfig set-connection-handler-prop \
--handler-name "LDAP Connection Handler" \
--set allow-start-tls:true \
--set key-manager-provider:JKS \
--set trust-manager-provider:JKS
```

2. Use `ldapsearch` to test StartTLS.

```
$ bin/ldapsearch -p 1389 --useStartTLS -b "" -s base "(objectclass=*)"
```

```
The server is using the following certificate:
Subject DN: CN=Server Cert, OU=Identity Proxy Certificate,
O=Example Company, L=Austin, ST=Texas, C=US
Issuer DN: EMAILADDRESS=whatever@example.com, CN=Cert Auth,
OU=My Certificate Authority, O=Example Company, L=Austin, ST=Texas, C=US
Validity: Thu Oct 29 10:29:59 CDT 2011 through Fri Oct 29 10:29:59 CDT 2012

Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes
```

```
dn:  
objectClass: ds-root-dse  
objectClass: top  
startupUUID: 6fa8f196-d112-40b4-b8d8-93d6d44d59ea
```

Authentication Mechanisms

The UnboundID Identity Proxy supports the use of both simple and Simple Authentication and Security Layer (SASL) authentication.

Simple Authentication

Simple authentication allows a client to identify itself to the Identity Proxy using the DN and password of the target user. Because the password is provided in the clear, simple authentication is inherently insecure unless the client communication is encrypted using a mechanism like SSL or StartTLS.

If both the DN and password of a simple bind request are empty (i.e., zero-length strings), then the server will process it as an anonymous bind. This will have no effect if the client is not already authenticated, but it can be used to destroy any previous authentication session and revert the connection to an unauthenticated state as if no bind had ever been performed on that connection.

Working with SASL Authentication

SASL (Simple Authentication and Security Layer, defined in RFC 4422) provides an extensible framework that can be used to add support for a range of authentication and authorization mechanisms. The UnboundID Identity Proxy provides support for a number of common SASL mechanisms.

Figure 8: Simple Authentication and Security Layer

Working with the SASL ANONYMOUS Mechanism

The ANONYMOUS SASL mechanism does not actually perform any authentication or authorization, but it can be used to destroy an existing authentication session. It also provides an option to allow the client to include a trace string, which can be used to identify the purpose of the connection. Because there is no authentication, the content of the trace string cannot be trusted.

The SASL ANONYMOUS mechanism is disabled by default but can be enabled if desired using the `dsconfig` tool. The SASL configuration options are available as an **Advanced menu** option using `dsconfig` in interactive mode.

The LDAP client tools provided with the Identity Proxy support the use of SASL ANONYMOUS. The optional "trace" SASL option may be used to specify the trace string to include in the bind request.

To Configure SASL ANONYMOUS

1. Use `dsconfig` to enable the SASL ANONYMOUS mechanism.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \  
--handler-name ANONYMOUS --set enabled:true
```

2. Use `ldapsearch` to view the root DSE and enter a trace string in the access log.

```
$ bin/ldapsearch --port 1389 --saslOption mech=ANONYMOUS \  
--saslOption "trace=debug trace string" --baseDN "" \  
--searchScope base "(objectclass=*)"
```

```
dn:  
objectClass: ds-root-dse  
objectClass: top  
startupUUID: 59bab79d-4429-49c8-8a88-c74a86792f26
```

3. View the access log using a text editor in `/ds/UnboundID-Proxy/logs` folder.

```
[26/Oct/2011:16:06:33 -0500] BIND RESULT conn=2 op=0 msgID=1 resultCode=0  
additionalInfo="trace='debug trace string'" etime=345.663  
clientConnectionPolicy="default"
```

Working with the SASL PLAIN Mechanism

SASL PLAIN is a password-based authentication mechanism which uses the following information:

- **Authentication ID.** Used to identify the target user to the server. It should be either "dn:" followed by the DN of the user or "u:" followed by a username. If the "u:"-style syntax is used, then an identify mapper will be used to map the specified username to a user entry. An authentication ID of "dn:" that is not actually followed by a DN may be used to request an anonymous bind.
- **Clear-text Password.** Specifies the password for the user targeted by the authentication ID. If the given authentication ID was "dn:", then this should be an empty string.
- **Optional Authorization ID.** Used to request that operations processed by the client be evaluated as if they had been requested by the user specified by the authorization ID rather than the authentication ID. It can allow one user to issue requests as if he/she had authenticated as another user. The use of an alternate authorization identity will only be allowed for clients with the `proxied-auth` privilege and the proxy access control permission.

Because the bind request includes the clear-text password, SASL PLAIN bind requests are as insecure as simple authentication. To avoid an observer from capturing passwords sent over the network, it is recommended that SASL PLAIN binds be issued over secure connections.

By default, the SASL PLAIN mechanism uses an Exact Match Identity Mapper that expects the provided username to exactly match the value of a specified attribute in exactly one entry

(for example, the provided user name must match the value of the `uid` attribute). However, an alternate identity mapper may be configured for this purpose which can identify the user in other ways (for example, transforming the provided user name with a regular expression before attempt to find a user entry with that transformed value).

LDAP clients provided with the server can use SASL PLAIN with the following SASL options:

- **authID**. Specifies the authentication ID to use for the bind. This must be provided.
- **authzID**. Specifies an optional alternate authorization ID to use for the bind.

To Configure SASL PLAIN

1. Use `dsconfig` to enable the SASL PLAIN mechanism.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
--handler-name PLAIN --set enabled:true
```

2. Use `ldapsearch` to view the root DSE using the authentication ID (`authid`) with the username `jdoe`. The `authid` option is required. Enter a password for the authentication ID.

```
$ bin/ldapsearch --port 1389 --saslOption mech=PLAIN \
--saslOption "authid=u:jdoe" --baseDN "" \
--searchScope base "(objectclass=*)"
Password for user 'u:jdoe':
```

Note: You can also specify the fully DN of the user when using the SASL PLAIN option:



```
$ bin/ldapsearch --port 1389 --saslOption mech=PLAIN \
--saslOption "authid=dn:uid=jdoe,ou=People,dc=example,dc=com" \
--baseDN "" --searchScope base "(objectclass=*)"
Password for user 'dn:uid=jdoe,ou=People,dc=example,dc=com':
```

```
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 59bab79d-4429-49c8-8a88-c74a86792f26
```

Working with the SASL CRAM-MD5 Mechanism

CRAM-MD5 is a password-based SASL mechanism that prevents exposure of the clear-text password by authenticating through the use of an MD5 digest generated from a number of elements, including the clear-text password, the provided authentication ID, and a challenge comprised of randomly-generated data. This ensures that the clear-text password itself is not transmitted, and the inclusion of server-generated random data protects against replay attacks.

During the CRAM-MD5 session, the client sends a bind request of type SASL CRAM-MD5. The Identity Proxy sends a response with a SASL "Bind in Progress" result code plus credential information that includes a randomly generated challenge string to the LDAP client. The client combines that challenge with other information, including the authentication ID and clear-text password and uses that to generate an MD5 digest to be included in the SASL credentials, along

with a clear-text version of the authentication ID. When the Identity Proxy receives the second request, it will receive the clear-text password from the target user's entry and generate the same digest. If the digest that the server generates matches what the client provided, then the client will have successfully demonstrated that it knows the correct password.

Note that although CRAM-MD5 does offer some level of protection for the password, so that it is not transferred in the clear, the MD5 digest that it uses is not as secure as the encryption used by SSL or StartTLS. As such, authentication mechanisms that use a clear-text password are more secure communication channel. However, the security that CRAM-MD5 offers may be sufficient for cases in which the performance overhead that SSL/StartTLS can incur. It is available for use in the UnboundID Identity Proxy because some clients may require it.

Also note that to successfully perform CRAM-MD5 authentication, the Identity Proxy must be able to obtain the clear-text password for the target user. By default, the Identity Proxy encodes passwords using a cryptographically secure one-way digest that does not allow it to determine the clear-text representation of the password. As such, if CRAM-MD5 is used, then the password storage schemes for any users that authenticate in this manner should be updated, so that they will use a password storage scheme that supports reversible encryption. It will be necessary for any existing users to change their passwords so that those passwords will be stored in reversible form. The reversible storage schemes supported by the Identity Proxy include:

- > 3DES
- > AES
- > BASE64
- > BLOWFISH
- > CLEAR
- > RC4

CRAM-MD5 uses an authentication ID to identify the user as whom to authenticate. The format of that authentication ID may be either "dn:" followed by the DN of the target user (or just "dn:" to perform an anonymous bind), or "u:" followed by a username. If the "u:"-style syntax is chosen, then an identity mapper will be used to identify the target user based on that username. The `dsconfig` tool may be used to configure the identity mapper to use CRAM-MD5 authentication.

The LDAP client tools provided with the Identity Proxy support the use of CRAM-MD5 authentication. The `authID` SASL option should be used to specify the authentication ID for the target user.

To Configure SASL CRAM-MD5

1. Use `dsconfig` to enable the SASL CRAM-MD5 mechanism if it is disabled. By default, the CRAM-MD5 mechanism is enabled.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \  
  --handler-name CRAM-MD5 --set enabled:true
```

2. For this example, create a password policy for CRAM-MD5 using a reversible password storage scheme, like 3DES.

```
$ bin/dsconfig create-password-policy \  
  --policy-name "Test UserPassword Policy" \  
  --storage-scheme 3DES
```

```
--set password-attribute:userpassword \
--set default-password-storage-scheme:3DES
```

3. Use `ldapmodify` to add the `ds-pwp-password-policy-dn` attribute to an entry to indicate the Test UserPassword Policy should be used for that entry. After you have typed the change, press **CTRL-D** to process the modify operation. This step assumes that you have already configured the Test Password Policy.

```
$ bin/ldapmodify
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Test UserPassword Policy,cn=Password Policies,cn=config

Processing MODIFY request for uid=jdoe,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=jdoe,ou=People,dc=example,dc=com
```

4. Use `ldapmodify` to change the `userPassword` to a reversible password storage scheme. The password storage scheme is specified in the user's password policy.

```
$ bin/ldapmodify
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: secret
```

An alternate method to change the `userPassword` attribute password storage scheme is to deprecate the old scheme and then bind as the user using simple authentication or SASL PLAIN. This action will cause any existing password encoding using a deprecated scheme to be re-encoded with the existing scheme.

5. Use `ldapssearch` to view the root DSE using the authentication ID (`authid`) option with the username `jdoe`. The `authid` option is required. Enter a password for the user.

```
$ bin/ldapssearch --port 1389 --sasloption mech=CRAM-MD5 \
--sasloption "authid=u:jdoe" --baseDN "" --searchScope base "(objectclass=*)"
Password for user 'u:jdoe':

dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 50567aa3-acd2-4106-a077-37a092275363
```

Working with the SASL DIGEST-MD5 Mechanism

The Identity Proxy supports the SASL DIGEST-MD5 mechanism, which is a stronger mechanism than SASL CRAM-MD5. Like the SASL CRAM-MD5 mechanism, the client authenticates to the Identity Proxy using a stronger digest of the authentication ID plus other information without exposing its clear-text password over the network.

During the DIGEST-MD5 session, the client sends a bind request of type SASL DIGEST-MD5. The Identity Proxy sends a response with a "Bind in Progress" message plus credential information that includes a random challenge string to the LDAP client. The client responds by sending a bind response that includes a digest of the server's random string, a separately generated client string, the authentication ID, the authorization ID if supplied, the user's clear-text password and some other information. The client then sends its second bind request. The Identity Proxy also calculates the digest of the client's credential. The Identity Proxy validates

the digest and retrieves the client's password. Upon completion, the server sends a success message to the client.

As with SASL CRAM-MD5, the client and the server must know the clear-text password for the user. By default, the Identity Proxy encodes passwords using a one-way storage scheme (Salted SHA-1) that stores an encoded representation of the password and does not allow it to determine the clear-text representation of the password. Any users requiring SASL DIGEST-MD5 authentication must use a password policy that supports two-way, reversible encryption, in which the password is encoded, stored, and later decoded when requested. The following password storage schemes are reversible:

- > 3DES
- > AES
- > BASE64
- > BLOWFISH
- > CLEAR
- > RC4

By default, SASL DIGEST-MD5 uses the Exact Match Identity Mapper, which returns a success result if the authorization ID is an exact match for the value of the `uid` attribute. Administrators can configure the SASL DIGEST-MD5 mechanism to use other identity mappers, such as the Regular Expression Identity Mapper or a custom Identity Mapper written using the UnboundID Server SDK.

In many ways, the DIGEST-MD5 SASL mechanism is very similar to the CRAM-MD5 mechanism. It avoids exposing the clear-text password through the use of an MD5 digest generated from the password and other information. It also supports the use of an alternate authorization ID to indicate that operations be processed under the authority of another user. Like CRAM-MD5, DIGEST-MD5 provides better security than mechanisms like SASL-PLAIN that send the clear-text password over an *unencrypted* channel.

DIGEST-MD5 is considered a stronger mechanism than CRAM-MD5, because it includes additional information in the digest that makes it harder to decipher, such as randomly-generated data from the client in addition to the server-provided challenge as well as other elements like a realm and digest URI. DIGEST-MD5 is also considered weaker than sending a clear-text password over an *encrypted* connection, because it requires the server to store passwords in reversible form, which can be a security risk. We recommend that CRAM-MD5 and DIGEST-MD5 be avoided unless required by clients.

The LDAP client tools provided with the Identity Proxy provide the ability to use DIGEST-MD5 authentication using the following properties:

- **authID**. Specifies the authentication ID for the target user, in either the "dn:" or "u:" forms. This property is required.
- **authzID**. Specifies an optional authorization ID that should be used for operations processed on the connection.
- **realm**. The realm in which the authentication should be processed. This may or may not be required, based on the server configuration.

- **digest-uri.** The digest URI that should be used for the bind. It should generally be "ldap://" followed by the fully-qualified address for the Identity Proxy. If this is not provided, then a value will be generated.
- **qop.** The quality of protection to use for the bind request. At present, only `auth` is supported (indicating that the DIGEST-MD5 bind should only be used for authentication and should not provide any subsequent integrity or confidentiality protection for the connection), and if no value is provided then `auth` will be assumed.

To Configure SASL DIGEST-MD5

1. Use `dsconfig` to enable the SASL DIGEST-MD5 mechanism if it is disabled. By default, the DIGEST-MD5 mechanism is enabled.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name DIGEST-MD5 --set enabled:true
```

2. Set up a reversible password storage scheme as outlined [Working with the SASL CRAM-MD5 Mechanism](#), steps 2–4, which is also required for DIGEST-MD5.
3. Use `ldapsearch` to view the root DSE using the authentication ID with the username `jdoue`. The `authid` option is required. Enter a password for the authentication ID.

```
$ bin/ldapsearch --port 1389 --saslOption mech=DIGEST-MD5 \
  --saslOption "authid=u:jdoue" --baseDN "" \
  --searchScope base "(objectclass=*)"
Password for user 'u:jdoue':
```

```
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 2188e4d4-c2bb-4ab9-8e1c-848e0168c9de
```

4. The user identified by the authentication ID requires the `proxied-auth` privilege to allow it to perform operations as another user.

```
$ bin/ldapmodify
dn: uid=jdoue,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

5. Use `ldapsearch` with the `authid` (required) and `authzid` option to test SASL DIGEST-MD5.

```
$ bin/ldapsearch --port 1389 --saslOption mech=DIGEST-MD5 \
  --saslOption authid=u:jdoue \
  --saslOption authzid=dn:uid=admin,dc=example,dc=com \
  --base "" --searchScope base "(objectclass=*)"
Password for user 'u:jdoue':
```

```
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 2188e4d4-c2bb-4ab9-8e1c-848e0168c9de
```

Working with the SASL EXTERNAL Mechanism

The SASL EXTERNAL mechanism allows a client to authenticate using information about the client, which is available to the server, but is not directly provided over LDAP. In the UnboundID Identity Proxy, SASL EXTERNAL requires the use of a client certificate provided during SSL or StartTLS negotiation. This is a very secure authentication mechanism that does not require the use of passwords, although its use on a broad scale is generally only feasible in environments with a PKI deployment.

Prior to the SASL EXTERNAL session exchange, the client should have successfully established a secure communication channel using SSL or StartTLS, and the client must have presented its own certificate to the server in the process. The SASL EXTERNAL bind request itself does not contain any credentials, and the server will use only the information contained in the provided client certificate to identify the target user.

The Identity Proxy's configuration settings for SASL EXTERNAL includes three important properties necessary for its successful operation:

- **certificate-validation-policy.** Indicates whether to check to see if the certificate presented by the client is present in the target user's entry. Possible values are:
 - **always** - Always require the peer certificate to be present in the user's entry. Authentication will fail if the user's entry does not contain any certificates, or if it contains one or more certificates and the certificate presented by the client is not included in the user's entry.
 - **ifpresent** - (Default) If the user's entry contains one or more certificates, require that one of them match the peer certificate. Authentication will be allowed to succeed if the user's entry does not have any certificates, but it will fail if the user's entry has one or more certificates and the certificate provided by the client is not included in the user's entry.
 - **never** - Do not look for the peer certificate to be present in the user's entry. Authentication may succeed if the user's entry does not contain any client certificates, or if the user's entry contains one or more certificates regardless of whether the provided certificate is included in that set.
- **certificate-attribute.** Specifies the name of the attribute that holds user certificates to be examined if the `ds-cfg-certificate-validation-policy` attribute has a value of `ifpresent` or `always`. This property must specify the name of a valid attribute type defined in the server schema. Default value is `userCertificate`. Note that LDAP generally requires certificate values to use the `"binary"` attribute modifier, so certificates should actually be stored in user entries using the attribute `"userCertificate;binary"` rather than just `"userCertificate"`.
- **certificate-mapper.** Specifies the certificate mapper that will be used to identify the target user based on the certificate presented by the client. For more information on certificate mappers, see [Configuring Certificate Mappers](#). The LDAP client tools provided with the Identity Proxy support the use of SASL EXTERNAL authentication. This mechanism does not require any specific SASL options to be provided (other than `mech=EXTERNAL` to indicate that SASL EXTERNAL should be used). However, additional arguments are required to use SSL or StartTLS, and to provide a keystore so that a client certificate will be available.

To Configure SASL EXTERNAL

1. Change to the server root directory.

```
$ cd /ds/UnboundID-Proxy
```

2. Determine the `certificate-validation-policy` property. If you do not need to store the DER-encoded representation of the client's certificate in the user's entry, skip to the next step.

If you select `Always`, you must ensure that the user's entry has the attribute present with a value. If you select `ifpresent`, you can optionally have the `userCertificate` attribute present. You can store the client's certificate in the user entry using `ldapmodify`.

```
$ bin/ldapmodify
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/client.der
```

3. If you have an attribute other than `userCertificate`, then specify it using the `certificate-attribute` property. You may need to update your schema to support the attribute.
4. Determine the `certificate-mapper` property. For more information on certificate mappers, see [Configuring Certificate Mappers](#).
5. Use `dsconfig` to enable the SASL EXTERNAL mechanism if it is disabled. By default, the SASL mechanism is enabled. For this example, set the `certificate-mapper` property to "Subject Attribute to User Attribute". All other defaults are kept.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
--handler-name EXTERNAL --set enabled:true \
--set "certificate-mapper:Subject Attribute to User Attribute"
```

6. Use `ldapsearch` to test SASL EXTERNAL.

```
$ bin/ldapsearch --port 1636 --useSSL \
--keyStorePath /path/to/clientkeystore \
--keyStorePasswordFile /path/to/clientkeystore.pin \
--trustStorePath /path/to/truststore \
--saslOption mech=EXTERNAL --baseDN "" \
--searchScope base "(objectClass=*)"
```

Working with the GSSAPI Mechanism

The SASL GSSAPI mechanism provides the ability to authenticate LDAP clients using Kerberos V, which is a single sign-on mechanism commonly used in enterprise environments. In these environments, user credentials are stored in the Kerberos key distribution center (KDC) rather than the Identity Proxy. When an LDAP client attempts to authenticate to the Identity Proxy using GSSAPI, a three-way exchange occurs that allows the client to verify its identity to the server through the KDC.

The Identity Proxy's support for GSSAPI is based on the Java Authentication and Authorization Service (JAAS). By default, the server will automatically generate a JAAS configuration that

should be appropriate for the most common use cases. For more complex deployments, it is possible for an administrator to supply a custom JAAS configuration that is most appropriate for that environment.

While the GSSAPI specification includes a provision for protecting client-server communication through integrity (in which the communication is not encrypted, but is signed so that it is possible to guarantee that it was not be altered in transit) or confidentiality (in which the communication is encrypted so that it cannot be examined by third-party observers), the Identity Proxy currently supports GSSAPI only for the purpose of authenticating clients but not for securing their communication with the server.

Preparing the Kerberos Environment for GSSAPI Authentication

To implement GSSAPI authentication in the Identity Proxy, it is assumed that you already have a working Kerberos V deployment in which the Identity Proxy and LDAP clients will participate. The process for creating such a deployment is beyond the scope of this documentation, and you should consult the documentation for your operating system to better understand how to construct a Kerberos deployment. However, there are a few things to keep in mind:

- It is recommended that the KDC be configured to use "aes128-cts" as the TKT and TGS encryption type, as this encryption type should be supported by all Java VMs. Some other encryption types may not be available by default in some Java runtime environments. In Kerberos environments using the MIT libraries, this can be achieved by ensuring that the following lines are present in the [libdefaults] section of the `/etc/krb.conf` configuration file on the KDC system:

```
default_tkt_enctypes = aes128-cts
default_tgs_enctypes = aes128-cts
permitted_enctypes = aes128-cts
```

- When a client uses Kerberos to authenticate to a server, the addresses of the target server and the KDC are used in cryptographic operations. It is important to ensure that all systems agree on the addresses of the Identity Proxy and KDC systems. It is therefore strongly recommended that DNS be configured so that the primary addresses for the KDC and Identity Proxy systems are the addresses that clients will use to communicate with them.
- Kerberos authentication is time-sensitive and if system clocks are not synchronized, then authentication may fail. It is therefore strongly recommended that NTP or some other form of time synchronization be used for all KDC, Identity Proxy, and client systems.

To authenticate itself to the Kerberos environment, the KDC should include both host and service principals for all Identity Proxysystems. The host principal is in the form "host/" followed by the fully-qualified address of the server system, and the service principal should generally be "ldap/" followed by the fully-qualified address (for example, "host/directory.example.com" and "ldap/directory.example.com", respectively). In a MIT Kerberos environment, the `kadmin` utility may be used to create these principals, as follows:

```
# /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin with password.
Password for kws/admin@EXAMPLE.COM:
kadmin: add_principal -randkey host/directory.example.com
WARNING: no policy specified for host/directory.example.com@EXAMPLE.COM;
        defaulting to no policy
Principal "host/directory.example.com@EXAMPLE.COM" created.
```

```
kadmin: ktadd host/directory.example.com
Entry for principal host/directory.example.com with kvno 3, encryption type AES-128
CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: add_principal -randkey ldap/directory.example.com
WARNING: no policy specified for ldap/directory.example.com@EXAMPLE.COM;
defaulting to no policy
Principal "ldap/directory.example.com@EXAMPLE.COM" created.
kadmin: quit
```

On each Identity Proxy system, the service principal for that instance must be exported to a keytab file, which may be accomplished using a command as follows:

```
# /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin with password.
Password for kws/admin@EXAMPLE.COM:
kadmin: ktadd -k /ds/UnboundID-Proxy/config/server.keytab ldap/directory.example.com
Entry for principal ldap/directory.example.com with kvno 4, encryption type AES-128
CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/ds/UnboundID-Proxy/config/
server.keytab.
kadmin: quit
```

Because this file contains the credentials that the Identity Proxy will use to authenticate to the KDC, it is strongly recommended that appropriate protection be taken to ensure that it is only accessible to the Identity Proxy itself (for example, by configuring file permissions and/or file system access controls).

Configuring the GSSAPI SASL Mechanism Handler

The GSSAPI SASL mechanism handler provides the following configuration options:

- **enabled.** Indicates whether the GSSAPI SASL mechanism handler is enabled for use in the server. By default, it is disabled.
- **kdc-address.** Specifies the address that the Identity Proxy should use to communicate with the KDC. If this is not specified, then the server will attempt to determine it from the underlying system configuration.
- **server-fqdn.** Specifies the fully-qualified domain name that clients will use to communicate with the Identity Proxy. If this is not specified, the server will attempt to determine it from the underlying system configuration.
- **realm.** Specifies the Kerberos realm that clients will use. If this is not specified, the server will attempt to determine it from the underlying system configuration.
- **kerberos-service-principal.** Specifies the service principal that the Identity Proxy will use to authenticate itself to the KDC. If this is not specified, the service principal will be "ldap/" followed by the fully-qualified server address (for example, ldap/directory.example.com).
- **keytab.** Specifies the path to the keytab file that holds the credentials for the Kerberos service principal that the Identity Proxy will use to authenticate itself to the KDC. If this is not specified, the server will use the system-wide keytab.
- **identify-mapper.** Specifies the identify mapper that the Identity Proxy will use to map a client's Kerberos principal to the entry of the corresponding user account in the server. In the default configuration, the server will use a regular expression identity mapper that will look for an entry with a uid value equal to the username portion of the Kerberos principal. For example, for a Kerberos principal of jdoe@EXAMPLE.COM, the identity mapper will perform an internal search with a filter of (uid=jdoe).

- **enable-debug.** Indicates whether the Identity Proxy should write debugging information about Kerberos-related processing (including JAAS processing) that the server performs. If enabled, this information will be written to standard error, which will appear in the `logs/server.out` log file.
- **jaas-config file.** Specifies the path to a JAAS configuration file that the server should use. If this is not specified, the server will generate a JAAS configuration file based on the values of the other configuration properties. It is recommended that this only be used in extraordinary circumstances in which the server-generated JAAS configuration is not acceptable.

Testing GSSAPI Authentication

Once the GSSAPI SASL mechanism handler has been enabled and configured in the Identity Proxy, then clients should be able to use GSSAPI to authenticate to the server using Kerberos. The `ldapsearch` tool provided with the Identity Proxy may be used to test this, with a command like:

```
$ bin/ldapsearch --hostname directory.example.com --port 389 \  
--saslOption mech=GSSAPI --saslOption authID=jdoe@EXAMPLE.COM \  
--baseDN "" --searchScope base "(objectClass=*)"
```

If the client already has a valid Kerberos session authenticated with a principal of `jdoe@EXAMPLE.COM`, then this command should make use of that existing session and proceed without requiring any further credentials. If there is no existing Kerberos session, then the `ldapsearch` command will prompt for the Kerberos password for that user (or it may be supplied using either the `--bindPassword` or `--bindPasswordFile` arguments).

The `--saslOption` command-line argument may be used to specify a number of properties related to SASL authentication, with values to that option be given in "name=value" format. When using SASL authentication, the `mech` property must always be used to specify the SASL mechanism to use, and `--saslOption mech=GSSAPI` indicates that the GSSAPI mechanism will be used. When the GSSAPI mechanism has been selected, then the following additional SASL options are available for use:

- **authid.** Specifies the authentication ID, which is the Kerberos principal for the user authenticating to the server. This option must always be provided when using GSSAPI.
- **authzID.** Specifies the authorization ID that should be used. At present, the Identity Proxy does not support the use of an alternate authorization identity, so this should either be omitted or identical to the value of the `authID` property.
- **kdc.** Specifies the address of the KDC that the client should use during the authentication processing. If this is not provided, the client will attempt to determine it from the system's Kerberos configuration.
- **realm.** Specifies the Kerberos realm that should be used. If this is not provided, the client will attempt to determine it from the system's Kerberos configuration.
- **protocol.** Specifies the protocol that the Identity Proxy uses for its service principal (i.e., the portion of the service principal that appears before the slash and fully-qualified server address). If this is not provided, a default protocol of "ldap" will be used.

- **useTicketCache.** Indicates whether the client should attempt to make use of a Kerberos ticket cache to leverage an existing Kerberos session, which may allow the client to authenticate to the server without the need to supply any additional credentials. If this is not provided, or if it is provided with a value of `TRUE`, then a ticket cache will be used if available. The use of a ticket cache may be disabled by providing this option with a value of `FALSE`.
- **requireCache.** Indicates whether to require the use of a ticket cache in order to leverage an existing Kerberos session rather than allowing the use of user-supplied credentials for authentication. By default, this will be assumed to have a value of `FALSE`, but if it is provided with a value of `TRUE`, then authentication will only be successful if the user already has an existing Kerberos session. This will be ignored if the `useTicketCache` option has been provided with a value of `FALSE`.
- **ticketCache.** Specifies the path to the file to use as the Kerberos ticket cache. If this is not provided, the default ticket cache file path will be assumed. This will be ignored if the `useTicketCache` option has been provided with a value of `FALSE`.
- **renewTGT.** Indicates whether to attempt to renew the user's ticket-granting ticket when authenticating with an existing Kerberos session. If this is not provided, a default value of `FALSE` will be used.
- **debug.** Indicates whether to write debug information about the GSSAPI authentication processing to standard error. By default, no debug information will be written, but it may be enabled with a value of `TRUE`.
- **configFile.** Used to specify the path to a JAAS configuration file that the client should use when performing GSSAPI processing. If this is not specified, then a default JAAS configuration file will be generated based on other properties.

These options are available for use with all tools supplied with the Identity Proxy which support SASL authentication.

Working with the UNBOUNDID-TOTP SASL Mechanism

The Identity Proxy supports a proprietary multifactor authentication mechanism that allows the server to use the Time-based One-Time Password (TOTP) algorithm, specified in RFC 6238. The TOTP algorithm is an extension of the Hash-based Message Authentication Code One-Time Password (HTOP) algorithm, specified in RFC 4226. The TOTP algorithm computes a temporary code using the current time and a secret key that is shared between the client app (e.g., Google Authenticator) and the server. When combined with a static password, a TOTP code can provide a means of multifactor authentication that offers dramatically better security than can be achieved using a static password by itself.

This proprietary security mechanism, UNBOUNDID-TOTP SASL, issues a bind request that includes at least an authentication ID and a TOTP code, but may also include an authorization ID and/or a static password. When the Identity Proxy receives such a bind request, it first uses the authentication ID to identify the user that is authenticating and then retrieves the shared secret from the user's entry (stored as a base32-encoded value in the `ds-auth-totp-shared-secret` operational attribute) and uses that in conjunction with the current time to generate a TOTP code. If that matches the code that the user entered, then that confirms that the client

knows the shared secret. If a static password was also provided, then the server will confirm that it matches what is stored in the `userPassword` attribute (or whatever password attribute is specified in the user's password policy). By default, the server will require the client to provide a static password, since without it, the client will only be performing single-factor authentication.

The Commercial Edition of the LDAP SDK for Java provides the necessary client-side support for the UNBOUNDID-TOTP SASL mechanism and provides a `com.unboundid.ldap.sdk.unboundidds.OneTimePassword` class to generate HOTP and TOTP codes for testing purposes.

Notes about the UnboundID-TOTP SASL Mechanism

The UnboundID-TOTP SASL mechanism supports some new features of interest that add extra security to your system:

- **Limiting the Reuse of the One-Time Password.** Although TOTP passwords are only valid for a limited period of time, it is possible that an individual observing an unencrypted TOTP authentication could replay the bind request in order to reuse the TOTP code as long as the server considers it valid. To avoid this, the `prevent-totp-reuse` property may be used to cause the server to store information in the user's entry about TOTP codes that have been used to successfully authenticate and may still be valid. Subsequent TOTP authentication attempts will then ensure that the provided TOTP code does not match a previously-used value.
- **Implementing the Validate TOTP Extended Operation.** The Identity Data Store supports a Validate TOTP Extended Operation, which validates the TOTP password without performing any authentication on the user. This feature is enabled by default. This is not needed for UNBOUNDID-TOTP SASL support and nor does it alter the authentication state of a connection in any way, but it may be useful for third-party applications to use TOTP as a type of "step-up" authentication mechanism or to add extra assurance about the identity of an already authenticated user.
- **Using Sensitive Attributes with the TOTP Shared Secret.** You can use a sensitive attribute definition to prevent clients from retrieving TOTP shared secrets from the server and to ensure that all shared secret changes occur over secure connections. Note that this sensitive attribute definition must be referenced from the `sensitive-attribute` property of a client connection policy or the global `sensitive-attribute` property to be enabled.

To Configure UNBOUNDID-TOTP SASL

1. Configure the server so that `ds-auth-totp-shared-secret` is a sensitive attribute that can only be set over a secure connection and cannot ever be retrieved from the server. Create the sensitive attribute and reference it from the global configuration using `dsconfig`.

```
$ bin/dsconfig create-sensitive-attribute \  
  --attribute-name ds-auth-totp-shared-secret \  
  --set attribute-type:ds-auth-totp-shared-secret \  
  --set allow-in-returned-entries:suppress \  
  --set allow-in-filter:reject \  
  --set allow-in-compare:reject \  
  --set allow-in-add:secure-only \  
  --set allow-in-modify:secure-only  
$ bin/dsconfig set-global-configuration-prop \  

```

```
--add sensitive-attribute:ds-auth-totp-shared-secret
```

2. Update a user entry so that it contains a `ds-auth-totp-shared-secret` attribute with a value that holds the base32-encoded shared secret that will be used for TOTP authentication. If you put the sensitive attribute in place, then you will need to do this over a secure connection, such as over SSL or StartTLS. There is no maximum limit to the length of the `ds-auth-totp-shared-secret` string, but there is a minimum length of 16 base32-encoded characters. Note that Google Authenticator requires a base32 string whose length is a multiple of 8, and it cannot include the padding character ("=").

```
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
add: ds-auth-totp-shared-secret
ds-auth-totp-shared-secret: ONSWG4TFORRW6ZDF
```

3. To test this feature, install a TOTP client. For this example, you can use the Google Authenticator app on your Android, iOS, and Blackberry mobile device. On the Google Authenticator app, choose the **Add Account** option to manually add an account. Enter a name and the same base32-encoded key that you assigned to the user in the previous step. The default account type is "Time Based"; do not choose "Counter Based". You should see an item with the name you selected and a six-digit code that will change every 30 seconds.



Note: The Google Authenticator app only needs to know the current time and the shared secret in order to compute the TOTP code. It does not require a Google account, nor does it require a data connection or the ability to perform network communication.

4. The Identity Proxy's tools provide support for the UNBOUNDID-TOTP SASL mechanism. You can run an LDAP search using the UNBOUNDID-TOTP SASL mechanism in the same way as any other SASL component.

```
$ bin/ldapsearch --saslOption mech=UNBOUNDID-TOTP \
--saslOption authID=u:user.0 \
--saslOption totpPassword=628094 \
--bindPassword password \
--baseDN "" \
--searchScope base \
"(objectClass=*)"
```

Working with the UNBOUNDID-DELIVERED-OTP SASL

The Identity Proxy now includes support for a new form of two-factor authentication, *UNBOUNDID-DELIVERED-OTP SASL*, which uses one-time passwords (OTPs) that are delivered to the end user through some out-of-band mechanism. Out of the box, the server provides support for e-mail (through the same SMTP external server approach used for email) and SMS (through the Twilio web service). The Server SDK also provides support for creating custom delivery mechanisms.

The process for authenticating using this new mechanism involves two steps:

- The client must first send a "deliver one-time password" extended request to the server. This request includes an authentication ID (either "dn:" followed by the DN or "u:" followed

by the username), the user's static password, and an optional set of allowed delivery mechanisms. If successful, this will cause the server to generate a one-time password, store it in the user's entry, and send it to the user through some mechanism.

- Once the user has received the one-time password, the client should perform an UNBOUNDID-DELIVERED-OTP SASL bind (which may be on the same connection or a different connection as was used to process the "deliver one-time password" extended operation). The credentials for this SASL mechanism include an authentication ID to identify the user, an optional authorization ID (if operations performed by the client should be authorized as a different user), and the one-time password that was delivered to them.

The static password is not included in the SASL bind request, but because the user must provide the static password in order to obtain the one-time password, it still qualifies as a form of multifactor authentication. Unlike UNBOUNDID-TOTP SASL, there is no need to have a shared secret between the client and the server, or any special client-side software to generate the one-time password, or a need to worry about whether the client and server clocks are roughly in sync.

To Configure the UNBOUNDID-DELIVERED OTP SASL

1. Add support for one or more OTP delivery mechanisms. For email, you first need to create an SMTP external server and associate it with the global configuration before you can create the delivery mechanism.

```
$ bin/dsconfig create-external-server \  
  --server-name "Intranet SMTP Server" \  
  --type smtp \  
  --set server-host-name:server.example.com  
  
$ bin/dsconfig set-global-configuration-prop \  
  --add "smtp-server:Intranet SMTP Server"  
  
$ bin/dsconfig create-otp-delivery-mechanism \  
  --mechanism-name E-Mail \  
  --type email \  
  --set enabled:true \  
  --set 'sender-address:otp@example.com' \  
  --set "email-address-attribute-type:mail" \  
  --set "message-subject:Your one-time password" \  
  --set "message-text-before-otp:Your one-time password: "
```

2. If you have a Twilio account, you can use it to configure the server to deliver one-time passwords over SMS.

```
dsconfig create-otp-delivery-mechanism \  
  --mechanism-name SMS \  
  --type twilio \  
  --set enabled:true  
  --set twilio-account-sid:xxxxx \  
  --set twilio-auth-token:xxxxx \  
  --set "sender-phone-number:xxxxx" \  
  --set phone-number-attribute-type:mobile \  
  --set "message-text-before-otp:Your one-time password: "
```

3. Once you have your OTP delivery mechanisms, you can configure the extended operation handler.

```
$ bin/dsconfig create-extended-operation-handler \  
  --handler-name "Deliver One-Time Password" \  
  --type deliver-otp \  
  --set enabled:true
```

```
--set enabled:true \
--set "identity-mapper:Exact Match" \
--set "password-generator:One-Time Password Generator" \
--set default-otp-delivery-mechanism:SMS \
--set default-otp-delivery-mechanism:E-Mail
```

4. Next, configure the SASL mechanism handler.

```
$ bin/dsconfig create-sasl-mechanism-handler \
--handler-name UNBOUNDID-DELIVERED-OTP \
--type unboundid-delivered-otp \
--set enabled:true \
--set "identity-mapper:Exact Match" \
--set "otp-validity-duration:5 minutes"
```

5. Make sure the server contains a user account with the account needed to deliver the one-time password to the user (i.e., a valid email address or mobile number).
6. Next, use the deliver one-time password extended operation to have the server generate and send a one-time password to the user. The Commercial Edition of UnboundID LDAP SDK contains support for the extended request and response needed to do this. In actual production deployments, you can create a web form to allow the user to enter the information and click a button. The server comes with a new deliver-one-time-password command-line tool that can achieve the same result.

```
$ bin/deliver-one-time-password \
--userName jdoe \
--promptForBindPassword \
--deliveryMechanism SMS
Enter the static password for the user:

Successfully delivered a one-time password via mechanism 'SMS' to '123-456-7890'
```

If processed successfully, you will receive a text as follows:

```
Your one-time password: 123456
```

7. Finally, authenticate to the server using the UNBOUNDID-DELIVERED-OTP SASL mechanism. The Commercial Edition of the LDAP SDK can help you accomplish this so that the user sees an interface. Or, you can use `ldapsearch` or some other tool to accomplish the same result.

```
$ bin/ldapsearch \
-o mech=UNBOUNDID-DELIVERED-OTP \
-o authID=u:jdoe \
-o otp=123456 \
-b '' \
-s base '(objectClass=*)' \
ds-supported-otp-delivery-mechanism
```

The search returns:

```
dn:
ds-supported-otp-delivery-mechanism: E-Mail
ds-supported-otp-delivery-mechanism: SMS
```

Configuring Pass-Through Authentication

Pass-through authentication (PTA) is a mechanism by which one Identity Proxy receives the bind request and can consult another Identity Proxy to authenticate the bind request.

Administrators can implement this functionality by configuring a PTA plug-in that enables the Identity Proxy to accept simple password-based bind operations.

To Configure Pass-Through Authentication

1. First, use `dsconfig` to define the external servers for the instances that will be used to perform the authentication. The bind DN is set to `uid=pass-through-user,dc=example,dc=com`, which is used to bind to the target LDAP server for simple authentication. The `verify-credentials-method` property ensures that a single set of connections for processing binds and all other types of operations is in place without changing the identity of the associated connection.

```
$ bin/dsconfig create-external-server \
--server-name "ds-with-pw-1.example.com:389" \
--type unboundid-proxy \
--set server-host-name:ds-with-pw-1.example.com \
--set server-port:389 \
--set "bind-dn:uid=pass-through-user,dc=example,dc=com" \
--set authentication-method:simple \
--set verify-credentials-method:retain-identity-control
```

2. Repeat step 1 so that you have multiple external servers in case one of them becomes unavailable.

```
$ bin/dsconfig create-external-server \
--server-name "ds-with-pw-2.example.com:389" \
--type unboundid-proxy \
--set server-host-name:ds-with-pw-2.example.com \
--set server-port:389 \
--set "bind-dn:uid=pass-through-user,dc=example,dc=com" \
--set authentication-method:simple \
--set verify-credentials-method:retain-identity-control
```

3. Create an instance of the pass-through authentication plug-in that will use the external server(s) as a source of authentication. Based on this configuration, the server will first try to process a local bind as the target user (`try-local-bind:true`). The `try-local-bind:true` together with the `override-local-password:true` means that if the local bind fails for any reason, then it will try sending the request to either `ds-with-pw-1.example.com:389` or `ds-with-pw-2.example.com:389` (`server-access-mode:round-robin`). If the bind succeeds against the remote server, then the local entry will be updated to store the password that was used (`update-local-password:true`). The number of connections to initially establish to the LDAP external server is set to 10 (`initial-connections:10`). The maximum number of connections maintained to the LDAP external server is 10 (`max-connections:10`).

```
$ bin/dsconfig create-plugin \
--plugin-name "Pass-Through Authentication" \
--type pass-through-authentication \
--set enabled:true \
--set server:ds-with-pw-1.example.com:389 \
--set server:ds-with-pw-2.example.com:389 \
--set try-local-bind:true \
--set override-local-password:true \
--set update-local-password:true \
--set server-access-mode:round-robin \
--set initial-connections:10 \
--set max-connections:10
```



Note:

The `try-local-bind` property works in conjunction with the `override-local-password` property. If `try-local-bind` is `true` and `override-local-password` is set to its default value of `false`, then the server attempts a local bind first. If it fails *because no password is set*, then it will forward the bind request to a remote server. If the password was set but still fails, the server will not send the request to the remote server.

If `try-local-bind` is `true` and `override-local-password` is `true`, then a local bind will be attempted. The server will forward the request to the remote server if the local bind fails for any reason.

Adding Operational Attributes that Restrict Authentication

The Identity Proxy provides a number of operational attributes that can be added to user entries in order to restrict the way those users can authenticate and the circumstances under which they can be used for proxied authorization. The operational attributes are as follows:

- **ds-auth-allowed-address.** Used to indicate that the user should only be allowed to authenticate from a specified set of client systems. Values should be specified as individual IP addresses, IP address patterns (using wildcards like `"1.2.3.*"`, CIDR notation like `"1.2.3.0/24"`, or subnet mask notation like `"1.2.3.0/255.255.255.0"`), individual DNS addresses, or DNS address patterns (using wildcards like `"*.example.com"`). If no allowed address values are present in a user entry, then no client address restrictions will be enforced for that user.
- **ds-auth-allowed-authentication-type.** Used to indicate that the user should only be allowed to authenticate in certain ways. Allowed values include `"simple"` (to indicate that the user should be allowed to bind using simple authentication) or `"sasl {mech}"` (to indicate that the user should be allowed to bind using the specified SASL mechanism, like `"sasl PLAIN"`). If no authentication type values are present in a user entry, then no authentication type restrictions will be enforced for that user.
- **ds-auth-require-secure-authentication.** Used to specify whether the user should be required to authenticate in a secure manner. If this attribute is present with a value of `"true"`, then that user will only be allowed to authenticate over a secure connection or using a mechanism that does not expose user credentials (e.g., the CRAM-MD5, DIGEST-MD5, and GSSAPI SASL mechanisms). If this attribute is present with a value of `"false"`, or it is not present in the user's entry, then the user will not be required to authenticate in a secure manner.
- **ds-auth-require-secure-connection.** Used to specify whether the user should be required to communicate with the server over a secure connection. If this attribute is present in a user entry with a value of `"true"`, then that user will only be allowed to communicate with the server over a secure connection (using SSL or StartTLS). If this attribute is present with a value of `"false"`, or if it is not present in the user's entry, then the user will not be required to use a secure connection.
- **ds-auth-is-proxyable.** Used to indicate whether the user can be used as the target of proxied authorization (using the proxied authorization v1 or v2 control, the intermediate client

control, or a SASL mechanism that allows specifying an alternate authorization identity). If this attribute is present in a user entry with a value of "required", then that user will not be allowed to authenticate directly to the server but instead will only be allowed to be referenced by proxied authorization. If this attribute is present with a value of "prohibited", then that user will not be allowed to be the target of proxied authorization but may only authenticate directly to the server. If this attribute is present with a value of "allowed", or if it is not present in the user's entry, then the user may authenticate directly against the server or be the target of proxied authorization.

- **ds-auth-is-proxyable-by.** Used to restrict the set of accounts that may target the user for proxied authorization. If this attribute is present in a user's entry, then its values must be the DNs of the users that can target the user for proxied authorization (as long as those users have sufficient rights to use proxied authorization). If it is absent from the user's entry, then any account with appropriate rights may target the user via proxied authorization.

Configuring Certificate Mappers

SASL EXTERNAL requires that a certificate mapper be configured in the server. The certificate mapper is used to identify the entry for the user to whom the certificate belongs. The Identity Proxy supports a number of certificate mapping options including:

- **Subject Equals DN.** The Subject Equals DN mapper expects the subject of the certificate to exactly match the DN of the associated user entry. This option is not often practical as certificate subjects (e.g., `cn=jdoe,ou=Client Cert,o=Example Company,c=Austin,st=Texas,c=US`) are not typically in the same form as an entry (e.g., `cn=jdoe,ou=People,o=Example Company`, or `uid=jdoe,ou=People,dc=example,dc=com`).
- **Fingerprint.** The Fingerprint mapper expects the user's entry to contain an attribute (`ds-certificate-fingerprint` by default, although this is configurable), whose values are the SHA-1 or MD5 fingerprints of the certificate(s) that they can use to authenticate. This attribute must be indexed for equality.
- **Subject Attribute to User Attribute.** The Subject Attribute to User Attribute mapper can be used to build a search filter to find the appropriate user entry based on information contained in the certificate subject. For example the default configuration expects the `cn` value from the certificate subject to match the `cn` value of the user's entry, and the `e` value from the certificate subject to match the `mail` value of the user's entry.
- **Subject DN to User Attribute.** The Subject DN to User Attribute mapper expects the user's entry to contain an attribute (`ds-certificate-subject-dn` by default, although this is configurable), whose values are the subjects of the certificate(s) that they can use to authenticate. This multi-valued attribute can contain the subjects of multiple certificates. The attribute must be indexed for equality.

Configuring the Subject Equals DN Certificate Mapper

The Subject Equals DN Certificate Mapper is the default mapping option for the SASL EXTERNAL mechanism. The mapper requires that the subject of the client certificate exactly

match the distinguished name (DN) of the corresponding user entry. The mapper, however, is only practical if the certificate subject has the same format as your Identity Proxy's entries.

To Configure the Subject Equals DN Certificate Mapper

- Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Subject Equals DN"
```

Configuring the Fingerprint Certificate Mapper

The Fingerprint Mapper causes the server to compute an MD5 or SHA-1 fingerprint of the certificate presented by the client and performs a search to find that fingerprint value in a user's entry (ds-certificate-fingerprint by default). The ds-certificate-fingerprint attribute can be added to the user's entry together with the ds-certificate-user auxiliary object class. For multiple certificates, the attribute can have separate values for each of the acceptable certificates. If you decide to use this attribute, you must index the attribute as it is not indexed by default.

The following example will use this certificate:

```
Alias name: client-cert
Creation date: Oct 29, 2011
Entry type: PrivateKeyEntry

Certificate chain length: 1 Certificate[1]:
Owner: CN=jdoe, OU=Client Cert, O=Example Company, L=Austin, ST=Texas, C=US
Issuer: EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate Authority,
O=Example Company, L=Austin, ST=Texas, C=US
Serial number: e19cb2838441dbcd
Valid from: Thu Oct 29 13:07:10 CDT 2011 until: Fri Oct 29 13:07:10 CDT 2012
Certificate fingerprints:
    MD5: 40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
    SHA1: 2A:89:71:06:1A:F5:DA:FF:51:7B:3D:2D:07:2E:33:BE:C6:5D:97:13
    Signature algorithm name: SHA1withRSA
    Version: 1
```

To Configure the Fingerprint Certificate Mapper

1. Create an LDIF file to hold a modification that adds the ds-certificate-user object class and ds-certificate-fingerprint attribute to the target user's entry.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: ds-certificate-user
-
add: ds-certificate-fingerprint
ds-certificate-fingerprint: 40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
```

2. Then, apply the change to the entry using ldapmodify:

```
$ bin/ldapmodify --filename add-cert-attr.ldif
```

```
dn: uid=jdoe,ou=People,dc=example,dc=com
ds-certificate-fingerprint:40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
```

3. Check that the attribute was added to the entry using `ldapsearch`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=jdoe)" \
ds-certificate-fingerprint
dn:uid=jdoe,ou=People,dc=example,dc=com
ds-certificate-fingerprint:40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
```

4. Create an index for the `ds-certificate-fingerprint` attribute. If the server is configured with multiple data backends, then the attribute should be indexed in each of those backends.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
--index-name ds-certificate-fingerprint --set index-type:equality
```

5. Use the `rebuild-index` tool to cause an index to be generated for this attribute.

```
$ bin/rebuild-index --task --baseDN dc=example,dc=com \
--index ds-certificate-fingerprint

[14:56:28] The console logging output is also available in
'/ds/UnboundID-Proxy/logs/tools/rebuild-index.log'
[14:56:29] Due to changes in the configuration, index
dc_example_dc_com_ds-certificate-fingerprint.equality is currently
operating in a degraded state and must be rebuilt before it can used
[14:56:29] Rebuild of index(es) ds-certificate-fingerprint started with 161 total
records to process
[14:56:29] Rebuild complete. Processed 161 records in 0 seconds
(average rate 1125.9/sec)
```

6. Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
--handler-name EXTERNAL \
--set "certificate-mapper:Fingerprint Mapper"
```

Configuring the Subject Attribute to User Attribute Certificate Mapper

The Subject Attribute to User Attribute Certificate Mapper maps common attributes from the subject of the client certificate to the user's entry. The generated search filter must match exactly one entry within the scope of the base DN(s) for the mapper. If no match is returned or if multiple matches are found, the mapping fails.

Given the subject of the client certificate:

```
Owner: CN=John Doe, OU=Client Cert, O=Example Company, L=Austin, ST=Texas, C=US
```

We want to match to the following user entry:

```
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: jdoe
givenName: John
sn: Doe
cn: John Doe
mail: jdoe@example.com
```

To Configure the Subject Attribute to User Attribute Certificate Mapper

- Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Subject Attribute to User Attribute"
```

Configuring the Subject DN to User Attribute Certificate Mapper

The Subject DN to User Attribute Certificate mapper expects the user's entry to contain an attribute (`ds-certificate-subject-dn` by default) whose values match the subjects of the certificates that the user can use to authenticate. The `ds-certificate-subject-dn` attribute can be added to the user's entry together with the `ds-certificate-user` auxiliary object class. The attribute is multi-valued and can contain the Subject DN's of multiple certificates. The certificate mapper must match exactly one entry, or the mapping will fail.

If you decide to use this attribute, you must add an equality index for this attribute in all data backends.

To Configure the Subject DN to User Attribute Certificate Mapper

1. Create an LDIF file to hold a modification that adds the `ds-certificate-user` object class and `ds-certificate-subject-dn` attribute to the target user's entry.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: ds-certificate-user
-
add: ds-certificate-subject-dn
ds-certificate-subject-dn:CN=John Doe,OU=Client Certificate,O=Example
Company,L=Austin,ST=Texas,C=US
```

2. Then, apply the change to the entry using `ldapmodify`:

```
$ bin/ldapmodify --filename add-cert-attr.ldif
```

3. Check that the attribute was added to the entry using `ldapsearch`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=jdoe)" \
  ds-certificate-subject-dn

dn: uid=jdoe,ou=People,dc=example,dc=com
ds-certificate-fingerprint:CN=jdoe, OU=Client Cert, O=Example Company,
L=Austin, ST=Texas, C=US
```

4. Create an index to the `ds-certificate-subject-dn` attribute.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name ds-certificate-subject-dn --set index-type:equality
```

5. Use the `rebuild-index` tool to ensure that the index is properly generated in all appropriate backends.

```
$ bin/rebuild-index --task --baseDN dc=example,dc=com \  
--index ds-certificate-subject-dn
```

```
[15:39:19] The console logging output is also available in  
'/ds/UnboundID-Proxy/logs/ tools/rebuild-index.log'  
[15:39:20] Due to changes in the configuration, index  
dc_example_dc_com_ds-certificate-subject-dn.equality is currently operating  
in a degraded state and must be rebuilt before it can used  
[15:39:20] Rebuild of index(es) ds-certificate-subject-dn started with 161 total  
records to process  
[15:39:20] Rebuild complete. Processed 161 records in 0 seconds  
(average rate 2367.6/sec)
```

6. Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \  
--handler-name EXTERNAL \  
--set "certificate-mapper:Subject DN to User Attribute"
```

Chapter

9

Managing the Identity Proxy

Once you have configured the UnboundID Identity Proxy, 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*](#)
- [*Types of Log Publishers*](#)
- [*Creating New Log Publishers*](#)
- [*About Log Compression*](#)
- [*About Log Signing*](#)
- [*Configuring Log Rotation*](#)
- [*Configuring Log Retention*](#)
- [*Setting Resource Limits*](#)
- [*Monitoring the Identity Proxy*](#)
- [*Using the Monitoring Interfaces*](#)
- [*Profiling Server Performance Using the Stats Logger*](#)
- [*Working with Alarms, Alerts, and Gauges*](#)
- [*Working with Administrative Alert Handlers*](#)
- [*Working with Virtual Attributes*](#)
- [*About the Server SDK*](#)

Managing Logs

The Identity Proxy 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 Identity Proxy 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 Identity Proxy, 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy (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 Identity Data Store and the Identity Proxy. 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see [Configuring Log Rotation and Configuring Log Retention](#).

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
--type file-based-access --publisher-name "Disconnect Logger" \
--set enabled:true \
--set "rotation-policy:24 Hours Time Limit Rotation Policy" \
--set "rotation-policy:Size Limit Rotation Policy" \
--set "retention-policy:File Count Retention Policy" \
--set log-connects:false \
--set log-requests:false --set log-results:false \
--set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:



```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the 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 Identity Proxy 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.

About Log Compression

The Identity Proxy supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Identity Proxy supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be properly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit Logger" \
--set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

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

To Validate a Signed File

The Identity Proxy provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

Note: If any validation errors occur, you will see a message similar to the one as follows:



```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

Configuring Log Rotation

The Identity Proxy allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Identity Proxy 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 Identity Proxy allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Identity Proxy 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 Identity Proxy 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 Identity Proxy's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy.** Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \  
--publisher-name "File-Based Access Logger" \  
--set "retention-policy:Free Disk Space Retention Policy"
```

Setting Resource Limits

You can set resource limits for the Identity Proxy 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 Identity Proxy. 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 Identity Proxy. 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 Identity Proxy if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the Identity Proxy

While the Identity Proxy is running, it generates a significant amount of information available through monitor entries. This section contains information about the following:

- Monitoring Server Status Using the status Tool
- About the Monitor Entries
- Using the Monitoring Interfaces
- Monitoring with JMX

Monitoring System Data Using the 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.

To Monitor Server Using the Status Tool

The UnboundID Identity Proxy 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 Identity Proxy 5.0.1.0
```

```

Java Version:          jdk-7u9

    --- Connection Handlers ---

Address:Port : Protocol : State
-----:-----:-----:-----
0.0.0.0:1689 : JMX       : Disabled
0.0.0.0:636  : LDAPS     : Disabled
0.0.0.0:9389 : LDAP      : Enabled

    --- LDAP External Servers ---

Server          : Status      : Score : LB Algorithm
-----:-----:-----:-----
localhost:389   : Available : 10     : dc_example_dc_com-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 : 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 Identity Proxy 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 Identity Proxy version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Identity Proxy work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Using the Monitoring Interfaces

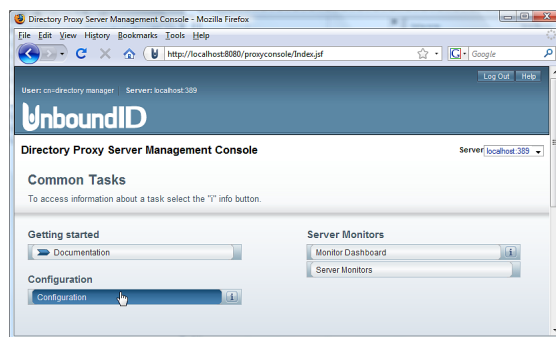
The UnboundID Identity Proxy exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the Management Console, JMX, over LDAP, using the LDAP SDK, and using SNMP.

Monitoring with the Management Console

UnboundID has developed a graphical web console for administrators to configure the identity data store. The console also provides a monitoring component that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Identity Proxy is running.
2. Open a browser to `http://hostname:8080/dsconsole/`. For information about installing the Management Console, see [Installing the 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 Identity Proxy 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 Management Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy 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`.

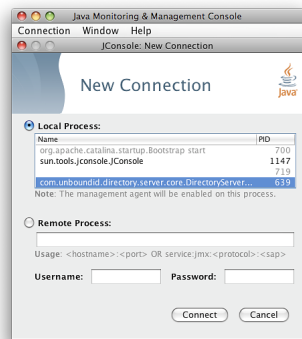
```
$ jconsole
```

Note: If SSL is configured on the JMX Connection Handler, you must specify the Identity Proxy jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

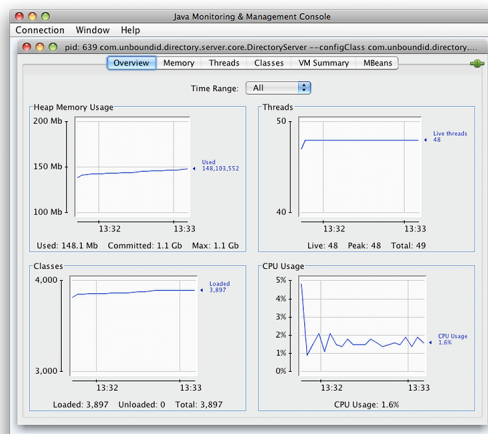


```
$ jconsole \
-J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
-J-Djavax.net.ssl.trustStorePassword=secret \
-J-Djava.class.path=$SERVER_ROOT/UnboundID-Proxy.jar:/Library/
Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/
jconsole.jar
```

2. On the **Java Monitoring & Management Console**, click **Local Process**, and then click the **PID** corresponding to the identity data store.



3. Review the resource monitoring information.



Monitoring the Identity Proxy Using JConsole

You can set up JConsole to monitor the Identity Proxy using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Identity Proxy using JConsole

1. Start the Identity Proxy.

```
$ 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
```

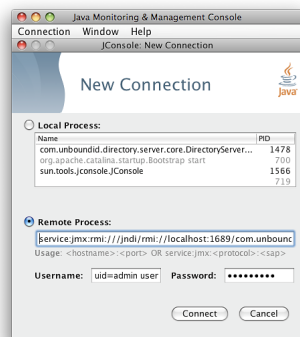
- Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ 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
```

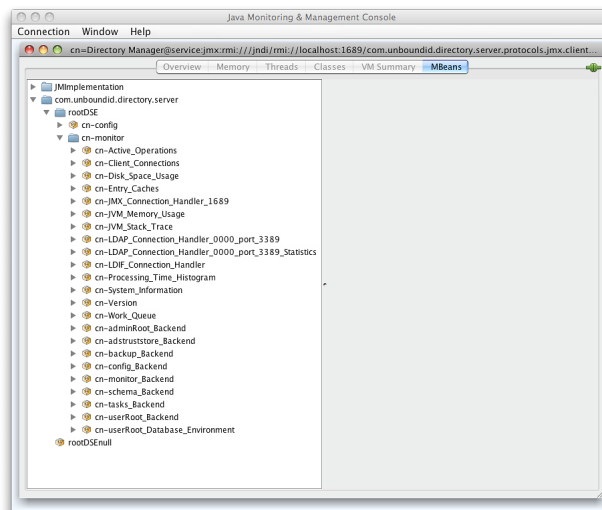
- On the **Java Monitoring & Management Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Identity Proxy.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

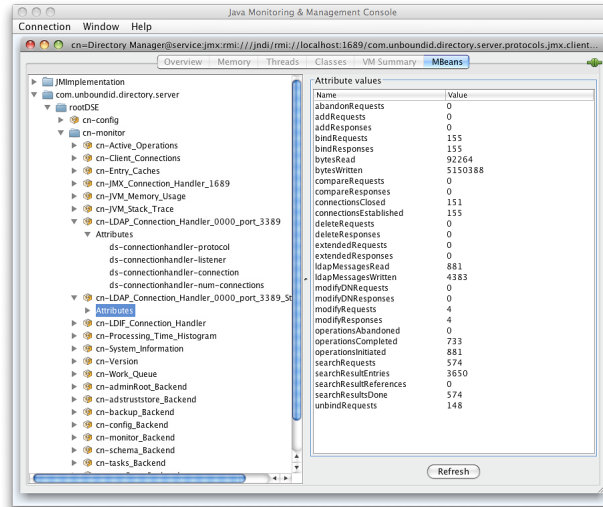
- In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



- Click **com.unboundid.directory.server**, and expand the **rootDSE** node and the **cn-monitor** sub-node.



- Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring over LDAP

The UnboundID Identity Proxy exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
--baseDN "cn=monitor" "(objectclass=*)"
```

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Identity Proxy as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Identity Proxy 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 Identity Proxy supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Identity Proxy 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, Identity Proxy instance, Identity Proxy, 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy's SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

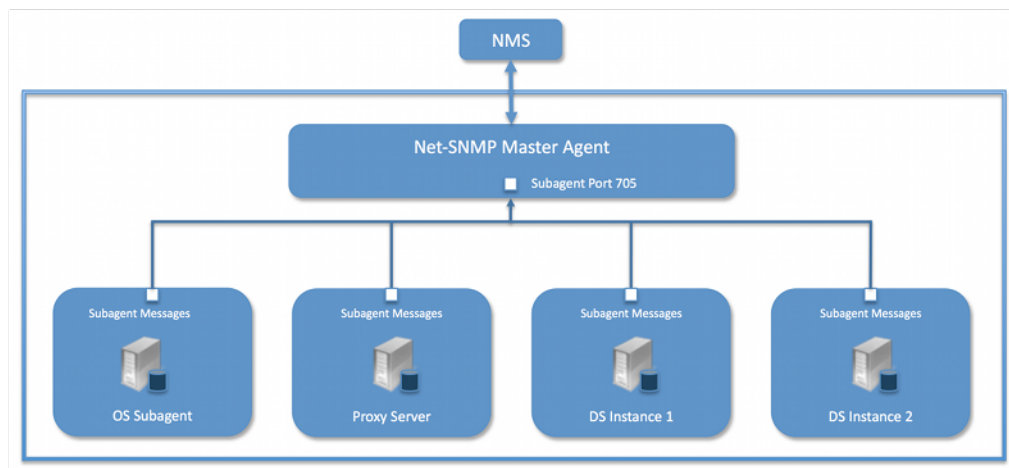


Figure 9: Example SNMP Deployment

One important note is that the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 the UnboundID Identity Proxy. 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy and the master agent. An eavesdropper might be able to listen in on the monitoring data sent by the UnboundID Identity Proxy. 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 Identity Proxy 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 Identity Proxy connects to, and statistics about the operations invoked by the Identity Proxy on those LDAP servers.
- **LDAP Statistics MIB.** Provides a collection of connection-oriented performance data that is based on a connection handler in the Identity Proxy. 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 Identity Data Store and the Identity Proxy, see the text files provided in the `resource/mib` directory below the server installation.

The Identity Proxy generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Profiling Server Performance Using the Stats Logger

The Identity Proxy ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-`

`interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Identity Proxy ships with the built-in '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 Identity Proxy 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 **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin
```

| Property | Value(s) |
|------------------------------|---|
| 1) description | Logs performance stats to a log file periodically. |
| 2) enabled | false |
| 3) local-db-backend-info | basic |
| 4) replication-info | basic |
| 5) entry-cache-info | - |
| 6) host-info | - |
| 7) included-ldap-application | If per-application LDAP stats is enabled, then stats will be included for all applications. |
| 8) log-interval | 1 s |
| 9) collection-interval | 200 ms |
| 10) suppress-if-idle | true |
| 11) header-prefix-per-column | false |
| 12) empty-instead-of-zero | true |
| 13) lines-between-header | 50 |
| 14) included-ldap-stat | active-operations, num-connections, op-count-and-latency, work-queue |
| 15) included-resource-stat | memory-utilization |
| 16) histogram-format | count |
| 17) histogram-op-type | all |

```

18) per-application-ldap-stats aggregate-only
19) ldap-changelog-info -
20) gauge-info none
21) log-file logs/dsstats.csv
22) log-file-permissions 640
23) append true
24) rotation-policy Fixed Time Rotation Policy, Size Limit
    Rotation Policy
25) retention-policy File Count Retention Policy

?) help
f) finish - apply any changes to the Periodic Stats Logger Plugin
a) hide advanced properties of the Periodic Stats Logger Plugin
d) display the equivalent dsconfig command lines to either re-create this
    object or only to apply pending changes
b) back
q) quit

Enter choice [b]:

```

7. Run the Identity Proxy. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate --help` to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.

To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in [To Enable the Stats Logger](#).
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.

4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.



Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to `memory-utilization` on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the Identity Proxy configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN `"cn=JVM Memory Usage,cn=monitor"` entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press **Enter** to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter `1073741824` (i.e., `1073741824 bytes = 1 gigabytes`).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
>>>> via creating 'Memory Usage' Custom Logged Stats

Property                                Value(s)
```

```

1)  description                -
2)  enabled                    true
3)  monitor-objectclass        ds-memory-usage-monitor-entry
4)  include-filter             -
5)  attribute-to-log           total-bytes-used-by-memory-consumers
6)  column-name                Memory Consumer Total (GB)
7)  statistic-type             raw
8)  header-prefix              -
9)  header-prefix-attribute    -
10)  regex-pattern             -
11)  regex-replacement         -
12)  divide-value-by           1073741824
13)  divide-value-by-attribute -
14)  decimal-format            #.##
15)  non-zero-implies-not-idle false

?)  help
f)  finish - create the new Custom Logged Stats
a)  hide advanced properties of the Custom Logged Stats
d)  display the equivalent dsconfig arguments to create this object
b)  back
q)  quit

```

Enter choice [b]:

The Custom Logged Stats was created successfully

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```

$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824

```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or 'server unavailable' (Indicator Gauge). Gauges generate alarms, when the gauge's severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the

actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Management Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Identity Proxy installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Identity Proxy is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

          --- Administrative Alerts ---
Severity : Time                : Message
-----:-----:-----
Info    : 11/Aug/2014         : A configuration change has been made in the Identity
        : 15:48:46 -0500     : Data Store:
        :                   : [11/Aug/2014:15:48:46.054 -0500]
        :                   : conn=17 op=73 dn='cn=Directory Manager,cn=Root
```

```

:      : DNSs,cn=config' authtype=[Simple] from=127.0.0.1
:      : to=127.0.0.1 command='dsconfig set-gauge-prop
:      : --gauge-name 'Cleaner Backlog (Number Of Files)'
:      : --set warning-value:-1'
Info   : 11/Aug/2014      : A configuration change has been made in the Identity
: 15:47:32 -0500 : Data Store: [11/Aug/2014:15:47:32.547 -0500]
:      : conn=4 op=196 dn='cn=Directory Manager,cn=Root
:      : DNSs,cn=config' authtype=[Simple] from=127.0.0.1
:      : to=127.0.0.1 command='dsconfig set-gauge-prop
:      : --gauge-name 'Cleaner Backlog (Number Of Files)'
:      : --set warning-value:0'
Error  : 11/Aug/2014      : Alarm [CPU Usage (Percent). Gauge CPU Usage (Percent)
: 15:41:00 -0500 : for Host System has
:      : a current value of '18.583333333333332'.
:      : The severity is currently OVERRIDDEN in the
:      : Gauge's configuration to 'CRITICAL'.
:      : The actual severity is: The severity is
:      : currently 'NORMAL', having assumed this severity
:      : Mon Aug 11 15:41:00 CDT 2014. If CPU use is high,
:      : check the server's current workload and make any
:      : needed adjustments. Reducing the load on the system
:      : will lead to better response times.
:      : Resource='Host System']
:      : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48 hours
Use the --maxAlerts and/or --alertSeverity options to filter this list

```

```

--- Alarms ---
Severity : Severity Start : Condition : Resource : Details
: Time : : : :
-----
Critical : 11/Aug/2014      : CPU Usage : Host System : Gauge CPU Usage (Percent) for
: 15:41:00 -0500 : (Percent) : : Host System
:      : : : : has a current value of
:      : : : : '18.785714285714285'.
:      : : : : The severity is currently
:      : : : : 'CRITICAL', having assumed
:      : : : : this severity Mon Aug 11
:      : : : : 15:49:00 CDT 2014. If CPU use
:      : : : : is high, check the server's
:      : : : : current workload and make any
:      : : : : needed adjustments. Reducing
:      : : : : the load on the system will
:      : : : : lead to better response times
Warning  : 11/Aug/2014      : Work Queue: Work Queue : Gauge Work Queue Size (Number
: 15:39:40 -0500 : Size : : of Requests) for Work Queue
:      : (Number of: : has a current value of '27'.
:      : Requests) : : The severity is currently
:      : : : : 'WARNING' having assumed this
:      : : : : severity Mon Aug 11 15:48:50
:      : : : : CDT 2014. If all worker
:      : : : : threads are busy processing
:      : : : : other client requests, then
:      : : : : new requests that arrive will
:      : : : : be forced to wait in the work
:      : : : : queue until a worker thread
:      : : : : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list

```

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the status tool or Web Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Data Store is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the `-alarmSeverity=indeterminate` option:

| --- Alarms --- | | | | |
|----------------|------------------|------------------|-------------|------------------------------|
| Severity | : Severity Start | : Condition | : Resource | : Details |
| | : Time | | | |
| ----- | | | | |
| Normal | : 26/Aug/2014 | : Startup Begun | : cn=config | : The Identity Data Store |
| | : 14:16:29 -0500 | | | : is starting. |
| | : | : | : | : |
| Indeterminate | : 26/Aug/2014 | : Replication | : not | : The value of gauge |
| | : 14:16:40 -0500 | : Latency | : available | : Replication Latency |
| | : | : (Milliseconds) | : | : (Milliseconds) could not |
| | : | : | : | : be determined. The |
| | : | : | : | : severity is INDETERMINATE, |
| | : | : | : | : having assumed this |
| | : | : | : | : severity Tue Aug 26 |
| | : | : | : | : 14:17:10 CDT 2014. |

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge's entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor" \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn: Gauge Replication Latency (Milliseconds)
gauge-name: Replication Latency (Milliseconds)
resource:
severity: indeterminate
summary: The value of gauge Replication Latency (Milliseconds) could not
        be determined. The severity is INDETERMINATE, having assumed
        this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
              class ds-replica-monitor-entry
...
```

Working with Administrative Alert Handlers

The UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 Identity Proxy 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 Identity Proxy provides dynamically generated attributes called virtual attributes for local Identity Proxy data. The proxy virtual attributes apply to a local proxy backend, such as `cn=config` or the Root DSE. If you want to have virtual attributes in entries for proxied requests, then they must be configured in the backend servers. Alternately, attributes may be inserted into those entries using proxy transformations. For more information about configuring proxy transformations, see “Configuring Proxy Transformations”.

For example, you can define a virtual attribute and assign it to the Root DSE as follows:

```
$ bin/dsconfig create-virtual-attribute \  
  --name defineDescriptionOnRootDSE --type user-defined \  
  --set enabled:true --set attribute-type:description \  
  --set filter:objectclass=ds-root-dse --set value:PrimaryProxy
```

If you search the Root DSE using the following LDAP search, you see that the description attribute now has the value `PrimaryProxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \  
  --bindPassword "" --port 5389 -- hostname localhost \  
  "objectclass=*" description  
  
dn:  
description:PrimaryProxy
```

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Identity Proxy. 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

10

Managing Monitoring

The UnboundID Identity Proxy also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces via the UnboundID® Metrics Engine, the Management Console, SNMP, JMX, and over LDAP. The Identity Proxy also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

Topics:

- [*The Monitor Backend*](#)
- [*Monitoring Disk Space Usage*](#)
- [*Monitoring with the Metrics Engine*](#)
- [*Monitoring Using SNMP*](#)
- [*Monitoring with the Management Console*](#)
- [*Monitoring with JMX*](#)
- [*Monitoring Using the LDAP SDK*](#)
- [*Monitoring over LDAP*](#)
- [*Profiling Server Performance Using the Stats Logger*](#)

The Monitor Backend

The Identity Proxy exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the UnboundID Metrics Engine, through SNMP, the Management Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the `bin/status` tool to display server component activity and state.

The list of all monitor entries can be seen using `ldapsearch` as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
--baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 10: Identity Proxy Monitoring Components

| Component | Description |
|--|---|
| Active Operations | Provides information about the operations currently being processed by the Identity Proxy. Shows the number of operations, information on each operation, and the number of active persistent searches. |
| Backends | Provides general information about the state of an Identity Proxy backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size. |
| Client Connections | Provides information about all client connections to the Identity Proxy. The client connection information contains a name followed by an equal sign and a quoted value (e.g., <code>connID="15"</code> , <code>connectTime="20100308223038Z"</code> , etc.) |
| Connection Handlers | Provides information about the available connection handlers on the Identity Proxy, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients. |
| Disk Space Usage | Provides information about the disk space available to various components of the Identity Proxy. |
| General | Provides general information about the state of the Identity Proxy, including product name, vendor name, server version, etc. |
| Index | Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload. |
| HTTP/HTTPS Connection Handler Statistics | Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code. |
| JVM Stack Trace | Provides a stack trace of all threads processing within the JVM. |
| LDAP Connection Handler Statistics | Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc. |

| Component | Description |
|---------------------------|---|
| Processing Time Histogram | Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc. |
| System Information | Provides general information about the system and the JVM on which the Identity Proxy is running, including system host name, operation system, JVM architecture, Java home, Java version, etc. |
| Version | Provides information about the Identity Proxy version, including build ID, version, revision number, etc. |
| Work Queue | <p>Provides information about the state of the Identity Proxy work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, number of busy worker threads, etc.</p> <p>A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the <code>ldapsearch</code> command for example, use the <code>--useAdministrativeSession</code> option. The requester must have the <code>use-admin-session</code> privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the <code>num-administrative-session-worker-threads</code> property in the work queue configuration.</p> |

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Identity Proxy components. It also provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

You can configure three thresholds for this monitor:

- Low space warning threshold.** This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Identity Proxy will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Identity Proxy will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- Low space error threshold.** This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low

space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- **Out of space error threshold.** This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the UnboundID Identity Proxy will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the Metrics Engine

The UnboundID Metrics Engine is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various UnboundID servers in a deployment. The Metrics Engine relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your Identity Platform system. For more information, see the *UnboundID Metrics Engine Administration Guide*.

Monitoring Key Performance Indicators by Application

The UnboundID Identity Proxy can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the UnboundID identity infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the UnboundID Metrics Engine. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Identity Proxy Configuration section of the *UnboundID Metrics Engine Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the Metrics Engine by metrics

having the 'application-name' dimension. See the documentation under `docs/metrics` of the Metrics Engine for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the Metrics Engine, you need to configure the servers you will be monitoring: UnboundID Identity Data Store, UnboundID Identity Proxy, and UnboundID Identity Data Sync. The Metrics Engine requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Identity Proxy, you can use the `dsconfig` tool to make configuration changes for the Metrics Engine. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the Metrics Engine

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Identity Proxy keeps a metric history so that the Metrics Engine can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the Metrics Engine within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the Metrics Engine.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the Metrics Engine. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Identity Proxy keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Identity Proxy and Identity Proxy as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the Metrics Engine notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the Metrics Engine API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the `separate-monitor-entry-per-tracked-application` property of the processing time histogram plugin to `true`. This property must be set to expose per-application monitoring information under `cn=monitor`. When the `separate-monitor-entry-per-tracked-application` property is set to `true`, then the `per-application-ldap-stats` property must be set to `per-application-only` on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Processing Time Histogram" \
--set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the `per-application-ldap-stats` property of the Stats Collector plugin to `per-application-only`:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \
--set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Identity Proxy that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the `application-name` dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the `included-user-base-dn` property to include the Directory Manager's full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
```

```
--criteria-name "Directory Manager" \  
--type simple \  
--set "included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config"
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \  
--set "tracked-application:Directory Manager"
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Identity Proxy since the Identity Proxy passes the application name through to the Identity Data Store in the intermediate client control. If a client of the Identity Proxy or Identity Data Store happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the `tracked-application` property.

Monitoring Using SNMP

The UnboundID Identity Proxy supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Identity Proxy 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, Identity Proxy instance, Identity Proxy, 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy'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.

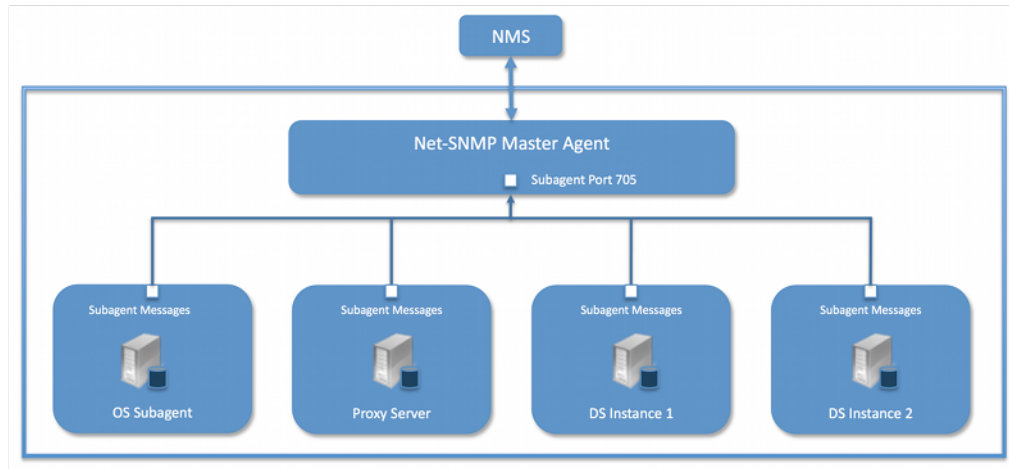


Figure 10: Example SNMP Deployment

One important note is that the UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 the UnboundID Identity Proxy. 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy and the master agent. An eavesdropper might be able to listen in on the monitoring data sent by the UnboundID Identity Proxy. 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 Identity Proxy 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 Identity Proxy connects to, and statistics about the operations invoked by the Identity Proxy on those LDAP servers.
- **LDAP Statistics MIB.** Provides a collection of connection-oriented performance data that is based on a connection handler in the Identity Proxy. A server typically contains 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 Identity Data Store and the Identity Proxy, see the text files provided in the `resource/mib` directory below the server installation.

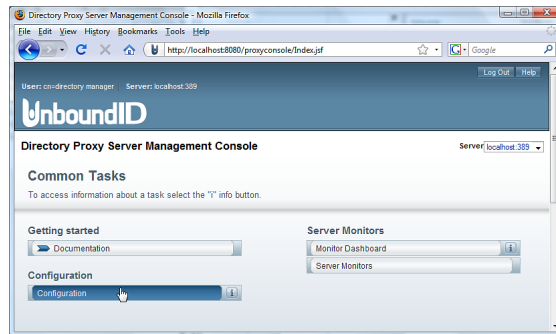
The Identity Proxy generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Management Console

UnboundID has developed a graphical web console for administrators to configure the identity data store. The console also provides a monitoring component that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Identity Proxy is running.
2. Open a browser to `http://hostname:8080/dsconsole/`. For information about installing the Management Console, see [Installing the 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 Identity Proxy 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 Management Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy 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`.

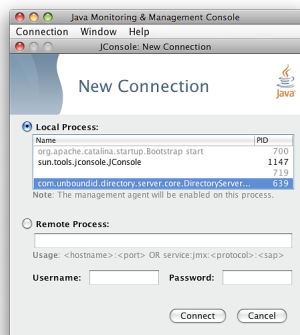
```
$ jconsole
```

Note: If SSL is configured on the JMX Connection Handler, you must specify the Identity Proxy jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

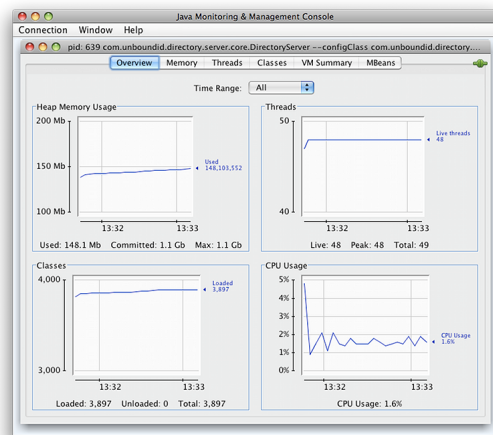


```
$ jconsole \  
-J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \  
-J-Djavax.net.ssl.trustStorePassword=secret \  
-J-Djava.class.path=$SERVER_ROOT/UnboundID-Proxy.jar:/Library/  
Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/  
jconsole.jar
```

2. On the **Java Monitoring & Management Console**, click **Local Process**, and then click the **PID** corresponding to the identity data store.



3. Review the resource monitoring information.



Monitoring the Identity Proxy Using JConsole

You can set up JConsole to monitor the Identity Proxy using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Identity Proxy using JConsole

1. Start the Identity Proxy.

```
$ 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
```

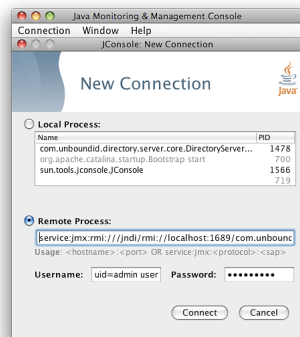
3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ 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
```

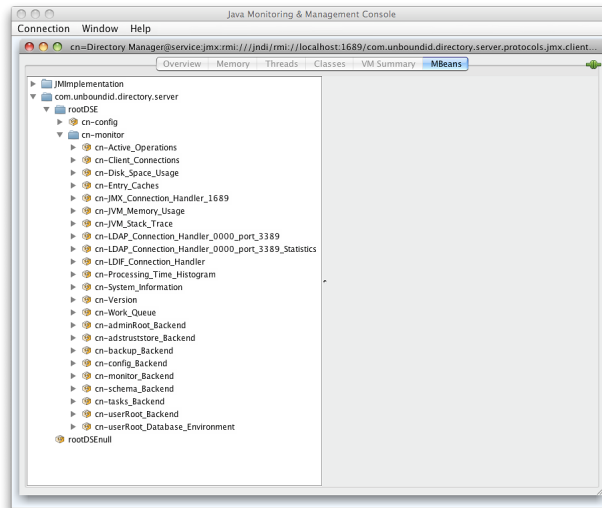
4. On the **Java Monitoring & Management Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Identity Proxy.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

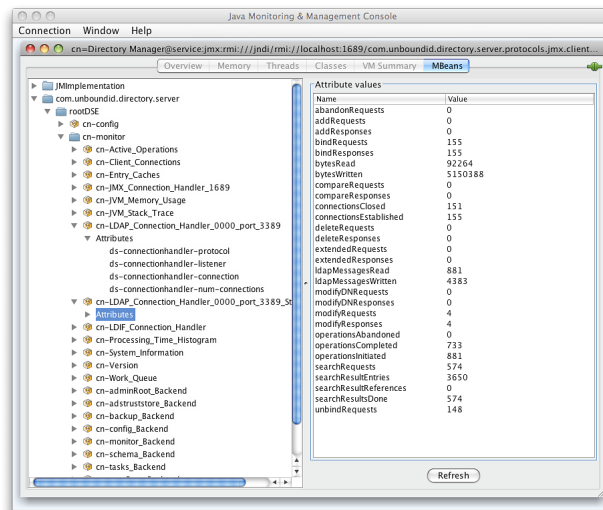
5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Identity Proxy as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Identity Proxy 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 over LDAP

The UnboundID Identity Proxy exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
--baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Identity Proxy ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Identity Proxy ships with the built-in "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 Identity Proxy 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 **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin
```

| Property | Value(s) |
|--------------------------|--|
| ----- | |
| 1) description | Logs performance stats to a log file periodically. |
| 2) enabled | false |
| 3) local-db-backend-info | basic |
| 4) replication-info | basic |
| 5) entry-cache-info | - |
| 6) host-info | - |

```

7)  included-ldap-application    If per-application LDAP stats is enabled,
                                then stats will be included for all
                                applications.
8)  log-interval                1 s
9)  collection-interval         200 ms
10) suppress-if-idle            true
11) header-prefix-per-column   false
12) empty-instead-of-zero      true
13) lines-between-header        50
14) included-ldap-stat          active-operations, num-connections,
                                op-count-and-latency, work-queue
15) included-resource-stat      memory-utilization
16) histogram-format            count
17) histogram-op-type          all
18) per-application-ldap-stats  aggregate-only
19) ldap-changelog-info         -
20) gauge-info                 none
21) log-file                    logs/dsstats.csv
22) log-file-permissions        640
23) append                     true
24) rotation-policy             Fixed Time Rotation Policy, Size Limit
                                Rotation Policy
25) retention-policy            File Count Retention Policy

?)  help
f)  finish - apply any changes to the Periodic Stats Logger Plugin
a)  hide advanced properties of the Periodic Stats Logger Plugin
d)  display the equivalent dsconfig command lines to either re-create this
    object or only to apply pending changes
b)  back
q)  quit

Enter choice [b]:

```

7. Run the Identity Proxy. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate --help` to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.

To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative

operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in [To Enable the Stats Logger](#).
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to `10s`, and the `suppress-if-idle` to `false`. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.



Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to `memory-utilization` on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```

dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037

```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the Identity Proxy configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the `objectclass` attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "`cn=JVM Memory Usage,cn=monitor`" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press **Enter** to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).

14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter **f** to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
>>>> via creating 'Memory Usage' Custom Logged Stats
```

| | Property | Value(s) |
|-----|---------------------------|--------------------------------------|
| 1) | description | - |
| 2) | enabled | true |
| 3) | monitor-objectclass | ds-memory-usage-monitor-entry |
| 4) | include-filter | - |
| 5) | attribute-to-log | total-bytes-used-by-memory-consumers |
| 6) | column-name | Memory Consumer Total (GB) |
| 7) | statistic-type | raw |
| 8) | header-prefix | - |
| 9) | header-prefix-attribute | - |
| 10) | regex-pattern | - |
| 11) | regex-replacement | - |
| 12) | divide-value-by | 1073741824 |
| 13) | divide-value-by-attribute | - |
| 14) | decimal-format | #.## |
| 15) | non-zero-implies-not-idle | false |

```
? ) help
f ) finish - create the new Custom Logged Stats
a ) hide advanced properties of the Custom Logged Stats
d ) display the equivalent dsconfig arguments to create this object
b ) back
q ) quit

Enter choice [b]:
```

The Custom Logged Stats was created successfully

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
--stats-name "Memory Usage" --type custom \
--set monitor-objectclass:ds-memory-usage-monitor-entry \
--set attribute-to-log:total-bytes-used-by-memory-consumers \
--set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
--set divide-value-by:1073741824
```

Chapter

11 Troubleshooting the Identity Proxy

This chapter provides the common problems and potential solutions that might occur when running UnboundID Identity Proxy. It is primarily targeted at cases in which the Identity Proxy 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*](#)
- [*Working with the Troubleshooting Tools*](#)
- [*Identity Proxy Troubleshooting Tools*](#)
- [*Troubleshooting Resources for Java Applications*](#)
- [*Troubleshooting Resources in the Operating System*](#)
- [*Common Problems and Potential Solutions*](#)

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 Identity Proxy 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 Identity Proxy (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 Identity Proxy 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 Identity Proxy. 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.

Identity Proxy Troubleshooting Tools

The UnboundID Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Identity Proxy. In many cases, obtaining the full benefit of these tools requires access to the Identity

Proxy source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the UnboundID Identity Proxy.

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 Identity Proxy is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the UnboundID Identity Proxy), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where `{processID}` is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The `"-l"` option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The `"-m"` option can be used to include native frames in the stack trace.

jmap

The `jmap` tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native `pmap` tool provided by many operating systems. As with the `jstack` tool, `jmap` can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the `jhat` tool
- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the `“:live”` portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Identity Proxy encounters an out of memory error (as a result of the `"-XX:+HeapDumpOnOutOfMemoryError"` JVM option) or from the use of the `jmap` command with the `"-dump"` option.

The `jhat` tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The `jhat` tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the `jhat` web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the `"-port"` option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The `jstat` tool is used to obtain a variety of statistical information from the JVM, much like the `vmstat` utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The `{interval}` option specifies the length of time in milliseconds between lines of output. The `{processID}` option specifies the process ID of the JVM used to run the Identity Proxy, which can be obtained by running `jps` as mentioned previously. The `{type}` option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails  
-XX:+PrintTenuringDistribution  
-XX:+PrintGCApplicationConcurrentTime  
-XX:+PrintGCApplicationStoppedTime  
-XX:+PrintGCDateStamps
```

To run the Identity Proxy 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 Identity Proxy 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.

Java Troubleshooting Tools (IBM JDK)

The UnboundID Identity Proxy 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 Identity Proxy. In some

cases, problems with the underlying system can be directly responsible for the issues seen with the Identity Proxy, 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy is consuming a significant amount of available CPU, it might be interfering with the ability of the Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy is a network-based application, it can be valuable to observe the network communication that it has with clients. The Identity Proxy 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 Identity Proxy 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 Identity Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see [Using the Collect Support Data Tool](#).



Important: Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be

passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The UnboundID Identity Proxy requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation that should be used. For example:

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

If this still does not work for some reason, then it can be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the `java` command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running `setup`, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Identity Proxy. 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 Identity Proxy prior to running the `setup` tool. This protects an existing Identity Proxy installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy, or reconfigure the Identity Proxy to reduce the amount of memory that it attempts to use.
- The Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 Identity Proxy 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 Identity Proxy configuration using an officially-supported tool like `dsconfig` or the 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy installation, so that they are owned by the user or role under which the server should run. For example, if the Identity Proxy 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 Identity Proxy was previously running but is no longer active, then the potential reasons include the following:

- The Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy does not appear to be accepting connections from clients, then potential reasons include the following:

- The Identity Proxy is not running.
- The underlying system on which the Identity Proxy is installed is not running.
- The Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.



Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Identity Proxy 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 Identity Proxy 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 Identity Proxy is running are not receiving TCP acknowledgement.
- If the system on which the Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy. 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 Identity Proxy'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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 Identity Proxy 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 identity proxies or load balancers or both to route requests away from the affected server. As of Identity Proxy version 3.1 or later, the Identity Proxy will indicate that it is in a degraded status and the Identity Proxy will route around it.
- If the Identity Proxy work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available

in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.

- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor` monitor entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
--bindDN "cn=directory manager" \
--bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3 msgID=4
  clientIP="10.8.4.21" authDN="cn=appl,ou=applications,dc=example,dc=com" base="dc=example,dc=com" scope=wholeSubtree filter="(description=expensive)" attrs="ALL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1 msgID=2
  clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root DNs,cn=config" base="cn=monitor" scope=wholeSubtree filter="(cn=active operations)" attrs="ALL"
num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name: com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif \
--defaultAdd --bindDN "cn=directory manager" \
--bindPassword password
```

Problems with the Management Console

If a problem arises when trying to use the 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 Identity Proxy is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Identity Proxy instance using the Management Console, then the access and error logs for that Identity Proxy instance might provide additional information about the underlying problem.

Problems with the Management Console: JVM Memory Issues

Console runs out of memory (PermGen). If you are running a Management Console for a UnboundID Identity Data Store while also running a console for the UnboundID Identity Proxy Management Console and an UnboundID Identity Data Sync 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 identity data stores. 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 identity proxy 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 Identity Proxy. 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 identity proxy 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 identity proxy instances to use the new account.
- You can use a protocol analyzer like `snoop` or `Wireshark`, to capture the password from the network communication.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter

12 Managing the SCIM Servlet Extension

The UnboundID Identity Proxy provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Topics:

- [*Overview of SCIM Fundamentals*](#)
- [*Creating Your Own SCIM Application*](#)
- [*Configuring SCIM*](#)
- [*Configuring Advanced SCIM Extension Features*](#)
- [*Configuring the Identity Access API*](#)
- [*Monitoring the SCIM Servlet Extension*](#)

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located at: <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 11: 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* | Yes (requires configuration for updated servers) |
| Bulk* | Yes |
| HTTP method overloading | Yes |
| Raw LDAP Endpoints** | Yes |

* denotes an optional feature of the SCIM protocol.

** denotes an UnboundID extension to the basic SCIM functionality.

About the Identity Access API

The UnboundID Identity Data Store, UnboundID Identity Proxy, and UnboundID Identity Data Sync support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access identity data store data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top --> person --> organizationalPerson --> inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the UnboundID Identity Data Store.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. UnboundID provides an open source SCIM SDK and Reference Implementation with which you can build a SCIM application.

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 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 Identity Proxy and allows for custom mappings between LDAP and SCIM data models. 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>1.5.0</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>.



Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring SCIM

This section discusses details about the UnboundID implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available from <http://www.simplecloud.info/>.

Before You Begin

To set up your SCIM servlet extension, the Identity Data Store provides a `dsconfig` batch file file, `scim-config-proxy.dsconfig`, located in the `<server-root>/config` directory. The script runs a series of commands that enables the HTTP Connection Handler and SCIM HTTP Servlet Extension, increases the level of detail logged by the HTTP Detailed Access Log Publisher, adds access controls to allow access to LDAP controls used by the SCIM servlet, adds support to the request processor for LDAP controls used by the SCIM servlet, and sets the subordinate base DN property of the root DSE so that SCIM requests can be authenticated using LDAP `uid` values. You should edit this `dsconfig` batch file before running the details of your deployment.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See [Managing the SCIM Schema](#) for more information.

Configuring the SCIM Servlet Extension

The Identity Proxy provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script, `scim-config-proxy.dsconfig`, located in the

`config` directory. The script runs a series of commands that enables the HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension.

When setting up the Identity Proxy to act as a SCIM server, some configuration must be performed on any identity data stores fronted by the Identity Proxy: enable the `entryDN` virtual attribute if also using the Identity Access API, and configure VLV indexes for `uid` (ascending) and `sn` (ascending) if search result pagination is desired.

To Configure the SCIM Servlet Extension

1. Before you enable the SCIM servlet extension, add access controls on each of the backend Identity Data Stores to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id ||
createTimestamp || modifyTimestamp")
(version 3.0;acl "Authenticated read access to operational attributes \
used by the SCIM servlet extension"; allow (read,search,compare)
userdn="ldap:///all";)'
```

2. On the Identity Proxy, enable the SCIM servlet extension by running the `dsconfig` batch file.

```
$ bin/dsconfig --batch-file config/scim-config-proxy.dsconfig
```

3. The `dsconfig` batch file must be edited to use the correct request processor name and base DN name(s) for the `set-request-processor-prop` and `set-root-dse-backend-prop` commands, respectively, as described in the "Configuring LDAP Control Support on All Request Processors" and "SCIM Servlet Extension Authentication" sections later in the chapter.

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

1. Enable the `ds-entry-checksum` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
--name ds-entry-checksum \
--set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
--remove 'global-aci:(targetattr="entryUUID || entryDN || ds-
entry-unique-id || createTimestamp || ds-create-time || modifyTimestamp || ds-update-
time")(version 3.0;acl "Authenticated read access to operational attributes used by
the SCIM servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

3. On the backend Data Store, make sure new access controls required by SCIM for read access to operational attributes are enabled with the following command. If this ACI is not present, issues will occur when a SCIM client tries to authenticate with a non-root DN.

```
$ bin/dsconfig set-access-control-handler-prop \
    --add 'global-aci:(targetattr="entryUUID || entryDN || ds-
entry-unique-id || createTimeStamp || ds-create-time || modifyTimestamp || ds-update-
time || ds-entry-checksum")(version 3.0;acl "Authenticated read access to operational
attributes used by the SCIM servlet extension"; allow (read,search,compare)
userdn="ldap:///all"'
```

Configuring LDAP Control Support on All Request Processors (Proxy Only)

You need to configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests. Change the request processor name that was provided as an example and repeat the command for all additional request processors.

To Configure LDAP Control Support on All Request Processors

- Use `dsconfig` to change the request processor name that was provided as an example and repeat the command for all additional request processors. Make sure to use your deployment's request processor name.

```
$ bin/dsconfig set-request-processor-prop \
    --processor-name dc_example_dc_com-req-processor \
    --add supported-control-oid:1.2.840.113556.1.4.319 \
    --add supported-control-oid:1.2.840.113556.1.4.473 \
    --add supported-control-oid:2.16.840.1.113730.3.4.9
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Identity Proxy. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, UnboundID provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

Enabling HTTPS Communications

If you want the SCIM HTTP connection handler to use SSL, which is mandated by the SCIM specification, you need to enable a Key Manager provider and Trust Manager provider.

To enable SSL during the Identity Proxy's initial setup, include the `--ldapsPort` and the `--generateSelfSignedCertificate` arguments with the `setup` command. If your server already has a certificate that you would like to use, set the `key-manager-provider` to the value you set when you enabled SSL in the Identity Proxy, or define a new key manager provider (see [Configuring HTTP Connection Handlers](#)).

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of `cn`.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the [UnboundID Server SDK documentation](#).

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Using HTTP Basic Authentication with Bare UID on the Directory Proxy Server

As discussed above, clients can authenticate to the SCIM extension using HTTP basic authentication and a bare UID value. However, when a SCIM extension is hosted by a Directory Proxy Server, the server needs to be explicitly configured with the names of subordinate base DNs to search. To do this, run the following command on the Directory Proxy Server for every base DN that may be accessed via SCIM. Make sure to specify your deployment's subordinate base DN.

```
$ bin/dsconfig set-root-dse-backend-prop \
```

```
--set subordinate-base-dn:dc=example,dc=com
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The `-k` (or `--insecure`) option is used to turn off `curl`'s verification of the server certificate, since the example Identity Proxy 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 Identity Proxy as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
-k "https://localhost:8443/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DN's of the Identity Proxy, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using `"dataType=binary"` in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the `<resource>` Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim ldap.ResourceMapper` class.

A `resource` element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the `<attribute>` Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.

- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
 - `simple`: specifies a simple, singular SCIM attribute.
 - `complex`: specifies a complex, singular SCIM attribute.
 - `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
 - `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the `<simple>` Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `decimal`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `simple` element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the `<complex>` Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the `<simpleMultiValued>` Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is `email`.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.

- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `simpleMultiValued` element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the `<complexMultiValued>` Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is `address`.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the `<subAttribute>` Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the `complexMultiValued` element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, `work` is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the [Mapping LDAP Entries to SCIM Using the SCIM-LDAP API](#) section.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in [Mapping LDAP Entries to SCIM Using the SCIM-LDAP API](#).

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DN's 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 `Identity Proxy`, meaning that a value is automatically provided by the Identity Proxy (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
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Identity Proxy, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Identity Proxy. 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 Identity Proxy 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 Identity Data Store Administration Guide.



Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM `dateTime` data type values to LDAP Generalized Time syntax values and vice-versa.

- `com.unboundid.scim ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the \$ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, 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>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.



Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP `DirectoryString` is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Identity Data Store and the Identity Proxy using the `OAuthTokenHandler` API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Identity Proxy already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "origin=scim" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191  
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191  
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "user-patch-404:26".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Identity Proxy LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.

- **include-ldap-base-dn.** Specifies a multi-valued property that lists the base DN's that will be exposed. If specified, only entries under these base DN's will be accessible. No parent-child relationships in the DN's are allowed here.
- **exclude-ldap-base-dn.** Specifies a multi-valued property that lists the base DN's that will not be exposed. If specified, entries under these base DN's will not be accessible. No parent-child relationships in the DN's are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM --set 'include-ldap-objectclass:*' \
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.



Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
  attributes=entryDN,uid,givenName,sn,entryUUID"
{"schemas":["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
```

```
"id": "56c9fd6b-f870-35ef-9959-691c783b7318",
"meta": {"lastModified": "2013-01-11T23:38:26.489Z",
"location": "https://example.com:443/v1/top/56c9fd6b-f870-35ef-9959-691c783b7318"},
"urn:unboundid:schemas:scim:ldap:1.0": {"givenName": ["Rufus"], "uid": ["user.1"],
"sn": ["Firefly"], "entryUUID": ["56c9fd6b-f870-35ef-9959-691c783b7318"],
"entrydn": "uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM reference implementation provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Identity Proxy automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM 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

```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.

- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'[dd/MM/yyyy:HH:mm:ss Z]'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter

13 Managing Server SDK Extensions

The UnboundID Identity Proxy provides support for any custom extensions that you create using the UnboundID® Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

Topics:

- [*About the Server SDK*](#)
- [*Available Types of Extensions*](#)

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Identity Proxy. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.



Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The UnboundID Server SDK provides support for creating a number of different types of extensions for UnboundID Server Products, including the UnboundID Identity Data Store, UnboundID Identity Proxy, and UnboundID Identity Data Sync. Some of those extensions include:

Cross-Product Extensions

- > Access Loggers
- > Alert Handlers
- > Error Loggers
- > Key Manager Providers
- > Monitor Providers
- > Trust Manager Providers
- > OAuth Token Handlers
- > Manage Extension Plugins

UnboundID Identity Data Store Extensions

- > Certificate Mappers
- > Change Subscription Handlers
- > Extended Operation Handlers
- > Identity Mappers
- > Password Generators
- > Password Storage Schemes
- > Password Validators
- > Plugins

- Tasks
- Virtual Attribute Providers

UnboundID Identity Proxy Extensions

- LDAP Health Checks
- Placement Algorithms
- Proxy Transformations

UnboundID Identity Data Sync Extensions

- JDBC Sync Sources
- JDBC Sync Destinations
- LDAP Sync Source Plugins
- LDAP Sync Destination Plugins
- Sync SourcesSync Destinations
- Sync Pipe Plugins

For more information on the UnboundID Server SDK, see the documentation available in the SDK build.

Chapter

14

Command-Line Tools

The UnboundID Identity Proxy provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Topics:

- [*Using the Help Option*](#)
- [*Available Command-Line Utilities*](#)
- [*Managing the tools.properties File*](#)
- [*Running Task-based Utilities*](#)

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing `--help` with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing `--help-subcommands`.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing `--help` with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```



Note: For detailed information and examples of the command-line tools, see the *UnboundID Identity Proxy Command-Line Tool Reference*.

Available Command-Line Utilities

The Identity Proxy provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 12: Command-Line Utilities

Command-Line Tools	Description
authrate	Perform repeated authentications against an LDAP identity data store, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more Identity Proxy backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
create-initial-proxy-config	Create an initial Identity Proxy configuration.
create-rc-script	Create a Run Control (RC) script that may be used to start, stop, and restart the Identity Proxy on UNIX-based systems.
dsconfig	View and edit the Identity Proxy configuration.
dsframework	Manage administrative server groups or the global administrative user accounts that are used to configure servers within server groups.

Command-Line Tools	Description
dsjavaproperties	Configure the JVM arguments used to run the Identity Proxy and associated tools. Before launching the command, edit the properties file located in <code>config/java.properties</code> to specify the desired JVM options and <code>JAVA_HOME</code> environment variable.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Identity Data Store.
enter-lockdown-mode	Request that the Identity Proxy enter lockdown mode, during which it only processes operations requested by users holding the <code>lockdown-mode</code> privilege.
global-index-size	Estimates the size in memory of one or more global indexes from the actual number of keys, the configured number of keys and the average key size.
ldap-diff	Compare the contents of two LDAP identity data store servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Identity Proxy.
ldapdelete	Perform LDAP delete operations in the Identity Proxy.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Identity Proxy.
ldappasswordmodify	Perform LDAP password modify operations in the Identity Proxy.
ldapsearch	Perform LDAP search operations in the Identity Proxy.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Identity Proxy leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in the Identity Proxy.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the UnboundID Identity Proxy. Extension bundles are installed from a zip archive or file system directory. The Identity Proxy will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Identity Proxy.
modrate	Perform repeated modifications against an LDAP identity data store.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
prepare-external-server	Prepare and a directory server for communication.
profile-viewer	View information in data files captured by the Identity Proxy profiler.
reload-index	Reload the contents of the global index.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Identity Proxy backend.
remove-defunct-server	Remove a server from this server's topology.
restore	Restore a backup of the Identity Proxy backend.
revert-update	Returns a server to the version before the last update was performed.

Command-Line Tools	Description
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP identity data store and modify each entry returned.
search-rate	Perform repeated searches against an LDAP identity data store.
server-state	View information about the current state of the Identity Proxy process.
setup	Perform the initial setup for the Identity Proxy instance.
start-proxy	Start the Identity Proxy.
status	Display basic server information.
stop-proxy	Stop or restart the Identity Proxy.
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Identity Data Store.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
summarize-config	Generate a configuration summary of either a remote or local Identity Proxy instance. By default, only basic components and properties will be included. To include advanced components, use the <code>--advanced</code> option.
uninstall	Uninstall the Identity Proxy.
update	Update the Identity Proxy to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the <code>update</code> tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the <code>revert-update</code> tool for information on reverting an update.
validate-ldif	Validate the contents of an LDIF file against the server schema.

Managing the tools.properties File

The UnboundID Identity Proxy supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool's arguments. Properties files are convenient when quickly testing the Identity Proxy in multiple environments.

The Identity Proxy supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the `--propertiesFilePath` option. You can override all of the Identity Proxy's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the `--propertiesFilePath /path/to/File` option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Identity Proxy also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```



Note: The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where `~` refers to the home directory, the server does not expand the `~` value when read from the properties file.

Specifying Default Properties Files

The Identity Proxy provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the `--propertiesFilePath` option. If you make changes to the `tools.properties` file, you do not need the `--propertiesFilePath` option. See the examples in the next section.

Evaluation Order Summary

The Identity Proxy uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the `--propertiesFilePath` option is used with no other options, the Identity Proxy takes its options from the specified properties file.
- If no options are used on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), the Identity Proxy searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Identity Proxy locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

```
$ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
  --propertiesFilePath bin/tools.properties "(objectclass=*)"
```

- Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Identity Proxy uses the specified properties file as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
  "(objectclass=*)"
```

- If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Identity Proxy attempts to locate any default `tools.properties` file in the following location:

```
<server-root>/config/tools.properties
```

Assume that you move your `tools.properties` file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

```
$ bin/ldapsearch "(objectclass=*)"
```

The Identity Proxy can be configured so that it does not search for a properties file by using the `--noPropertiesFile` option. This option tells the Identity Proxy to use only those options specified on the command line. The `--propertiesFilePath` and `--noPropertiesFile` options are mutually exclusive and cannot be used together.

- If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Identity Proxy has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, `bin/start-proxy`, `bin/start-proxy` and others. All task-based utilities require the `--task` option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 13: Task-based Utilities

Option	Description
<code>--task</code>	Indicates that the tool is invoked as a task. The <code>--task</code> argument is required. If a tool is invoked as a task without this <code>--task</code> argument, then a warning message will be displayed stating that it must be used. If the <code>--task</code> argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.
<code>--start</code>	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
<code>--dependency</code>	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
<code>--failedDependencyAction</code>	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: <code>PROCESS</code> , <code>CANCEL</code> , <code>DISABLE</code> . If not specified, the default value is <code>CANCEL</code> . This option can be used multiple times in a single command.

Option	Description
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.