UnboundID® Synchronization Server Administration Guide

Version 3.5.1

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This document constitutes an unpublished, copyrighted work and contains valuable trade secrets and other confidential information belonging to UnboundID Corporation.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. UNIX is a registered trademark in the United States and other countries, licenses exclusively through The Open Group. 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 2012 UnboundID Corporation

All Rights Reserved

Published: 2012-09-13

Contents

Preface..... ix
Purpose of This Guide.....ix
Audience.....ix
Related Documentation.....ix
Document Conventions.....x

Chapter 1: Introduction..... 1

Overview of the Synchronization Server.....2
The Synchronization Problem.....2
The UnboundID Advantage.....3
Common Synchronization Use Cases.....4
Use Case: Synchronization during Directory Server Migrations.....4
Advanced Replication.....5
Use Case: Synchronization with Active Directory.....8
Use Case: Synchronizing Realistic Test Environments.....9
Use Case: Synchronizing with Relational Databases.....9
Use Case: Synchronizing through Proxy Servers.....9
Synchronization Server: How It Works.....10
Point-to-Point Bidirectional Synchronization.....10
Synchronization Architecture.....11
Change Tracking.....11
Monitoring and Alerts.....12
Logging.....12
Synchronization Modes of Operation.....13
Standard Synchronization Mode.....13
Notification Synchronization Mode.....13
Sync Operations.....14
Data Transformations.....14
Resync.....14
Real-Time Synchronization.....15
About the Sync Retry Mechanism.....15
Configuration Model.....17
Sync Control Flow Scenarios.....19
A Synchronization Example.....21
Available Tools Summary.....21
Summary.....22

Chapter 2: Installing the Synchronization Server.....23

Before You Begin.....24
Supported Operating Platforms.....24
Software Requirements: Java 6.....24
About the Server Installation Modes.....25
Installing the UnboundID Synchronization Server in Interactive Mode.....25
To Install the Synchronization Server in Interactive Mode.....25
Installing the UnboundID Synchronization Server in Non-Interactive Mode.....28
To Install the Synchronization Server in Non-Interactive Mode.....28
Installing the Synchronization Server with a Truststore in Non-Interactive Mode.....28
To Install the Synchronization Server with a Truststore in Non-Interactive Mode.....28

Running the Server....29
To Start the Synchronization Server....29
To Start the Synchronization Server with Global Sync Disabled....29
To Run the Server as a Foreground Process....29
To Start the Server at Boot Time....30
Stopping the Directory Server....30
To Stop the Server....30
To Schedule a Server Shutdown....31
To Restart the Server....31
To Restart the Synchronization server using an Internal Restart....31
About the Server Uninstallation Modes....31
To Uninstall the Server in Interactive Mode....32
To Uninstall the Server in Non-Interactive Mode....33
To Uninstall Selected Components in Non-Interactive Mode....33
Installing the Sync Management Console....34
To Install the Sync Management Console Out of the Box....34
To Log into the Sync Management Console....36
To Uninstall the Sync Management Console....37
To Upgrade the Sync Management Console....37
Updating the Synchronization Server....38
To Update the Directory Proxy Server....38
Reverting an Update....39
Installing a Redundant Failover Server....40
To Install a Redundant Server....40
Removing a Redundant Server....41
To Remove a Redundant Server....41

Chapter 3: Configuring the Synchronization Server....43

Pre-Deployment Checklist....44
External Servers....44
Sync Pipes....44
Sync Classes....44
Creating Administrators....46
To Create an Administrator....46
About the Configuration Tools....46
About the Sync User Account....47
Configuring the Synchronzation Server in Standard Mode....48
Assumptions....48
Configuring the Synchronization using create-sync-pipe-config....49
Configuring the Synchronization Server Using the Management Console....55
Configuring the External Servers Using the Management Console....55
Configuring the Sync Pipe Using the Management Console....59
Configuring the Sync Class Using the Management Console....63
Starting the Global Sync Configuration Using the Management Console....68
Configuring the Server Using dsconfig....69
Using dsconfig in Interactive Command-Line Mode....69
Using dsconfig Interactive Mode: Viewing Object Menus....69
Using dsconfig in Non-Interactive Mode....70
Using dsconfig Batch Mode....72
Configuring the Synchronization Server Using dsconfig....72
To Configure the Synchronization Server Using dsconfig Interactive....72
Configuring Server Groups Using dsconfig Interactive....73
Configuring External Servers Using dsconfig Interactive....74
Configuring the Sync Source Using dsconfig Interactive....74
Configuring the Sync Destination Using dsconfig Interactive....75

Configuring a Sync Pipe Using dsconfig Interactive....76
Configuring the Sync Class Using dsconfig Interactive....77
Starting the Global Sync Configuration Using dsconfig Interactive....78
Generating a Summary of Configuration Components....78
To Generate a Summary of Configuration Components....78
Preparing the Synchronization Server for External Server Communication....81
To Prepare the Synchronization Server for External Server Communication....81
Preparing External Servers: If the Admin Does Not Have Root Access on DSEE External Servers....82
To Set Up the DSEE External Servers....82
Using Resync on the Synchronization Server....84
Testing Attribute and DN Maps Using Resync....86
Verifying the Synchronization Configuration Using Resync....86
Populating an Empty Sync Destination Topology Using Resync....87
Populating an Empty Sync Destination Topology Using translate-ldif....88
Setting the Synchronization Rate Using Resync....88
Synchronizing a Specific List of DNs....89
Controlling Real Time Synchronization....90
About the Realtime-Sync Tool....90
Starting Real Time Synchronization Globally....91
Pausing Synchronization....91
Setting Startpoints....92
Scheduling a Realtime Sync as a Task....94
Configuring Attribute Maps....94
Configuring an Attribute Map Using dsconfig Interactive....95
Configuring an Attribute Mapping Using dsconfig Interactive....95
Configuring an Attribute Mapping Using dsconfig Non-Interactive....96
Configuring the Directory Server Backend for Synchronizing Deletes....97
To Configure the Changelog-Deleted-Entry-Include-Attribute Property....97
To Synchronize Deletes on Sun DSEE Endpoints....97
Configuring DN Maps....98
Configuring a DN Map Using dsconfig Interactive....98
Configuring a DN Map Using dsconfig Non-Interactive....99
Configuring Fractional Replication....100
To Configure Fractional Replication....100
Managing Failover Behavior....102
Conditions that Trigger Immediate Failover....103
Failover Server Preference....103
Configuration Properties that Control Failover Behavior....105
max-operation-attempts....106
response-timeout....106
max-failover-error-code-frequency....107
max-backtrack-replication-latency....107
Managing Synchronization Server Extensions....108
To Run the Manage-Extension Tool....108

Chapter 4: Syncing with Active Directory Systems....109

Overview....110
Before You Begin....110
Configuring Active Directory Synchronization....110
To Configure Active Directory Synchronization....111
To Prepare the External Servers....112
To Configure the Sync Pipes and its Sync Classes....113
To Configure the Password Encryption Component....115
Installing the UnboundID Password Sync Agent....116
Supported Platforms....117

Before You Install the Password Sync Agent....117
To Install the Password Sync Agent....118
To Upgrade the Password Sync Agent (restart optional)....118
To Uninstall the Password Sync Agent....119
Manual Configuration for Advanced Users....119

Chapter 5: Syncing with Relational Databases....121

Overview....122
About the Server SDK....122
About the DBSync Process....123
About the DBSync Example....124
Example DS Entries....124
About the Overall DBSync Configuration Process....125
Downloading the Software Packages....126
Creating the JDBC Extension....126
About Groovy....127
Implementing a JDBC Sync Source....127
Implementing a JDBC Sync Destination....129
Configuring the Database for Synchronization....129
Pre-Configuration Checklist....131
General Tips When Syncing to a Database Destination....132
Configuring the Directory-to-Database Sync Pipe....134
Step 1. Creating the Directory-to-Database Sync Pipe....134
Step 2. Configuring the Sync Pipe and Sync Classes....137
Step 3. Fine-Tuning the Sync Classes....139
Step 4. Configuring the Attribute Mappings....143
Step 5. Run the Resync Tool to Test the Configuration....146
Step 6. Set the Startpoint in the Change Log....146
Step 7. Run the Resync Tool to Populate Data at the Destination Endpoint....146
Step 8. Start the Sync Pipe....146
Step 9. Debugging the Configuration....147
General Tips When Syncing from a Database Source....149
Configuring the Database-to-Directory Sync Pipe....149
To Create the Database-to-Directory Sync Pipe....150
Synchronizing a Specific List of Database Elements Using Resync....150
To Synchronize a Specific List of Database Elements Using Resync....151

Chapter 6: Syncing Through Proxy Servers.... 153

Features....154
How It Works....154
About the Get Changelog Batch Request and Get Server ID Controls....155
About the Directory Server and Directory Proxy Server Tokens....156
Change Log Tracking in Entry-Balancing Deployments....157
About the Overall Sync-through-Proxy Configuration Process....158
About the Sync-Through-Proxy Configuration Example....158
Configuring the Example Source Proxy Deployment....159
Configuring the Directory Servers....159
Configuring the Example Destination Proxy Deployment....163
To Configure the Synchronization Server....165
To Confirm the Proxy Server and Use-Changelog-Batch-Request Properties....167
To Run Prepare-External-Server on the Backend Set of Directory Servers....168
To Test and Start the Configuration....168
Indexing the LDAP Changelog....169

To Configure Changelog Indexing....170

Chapter 7: Configuring Notification Mode....171

About Notification Mode....172
Notification Mode Architecture....173
Sync Source Requirements....174
Failover Capabilities....174
Standard Administration and Monitoring Capabilities....175
Notification Sync Pipe Change Flow....175
About the Notification Mode Configuration....176
Create-Sync-Pipe-Config....176
No Resync....176
LDAP Change Log Features Required for Notifications....177
About the Server SDK and LDAP SDK....179
Server SDK Updates....179
LDAP SDK Updates....180
Important Design Questions....180
Implementing the Custom Server Extension....180
General Tips When Implementing Your Extension....181
Configuring the Notification Sync Pipe....182
General Tips When Configuring Your Sync Classes....182
Step 1. Creating the Notification Sync Pipe....183
Step 2. Configuring the Sync Pipe and Sync Classes....187
Step 3. Configure Attribute and DN Mappings....188
Step 4. Configure Advanced Properties....188
Step 5. Set the Startpoint in the Change Log....188
Step 6. Start the Sync Pipe....188
Step 7. Debugging the Configuration....188
Access Control Filtering on the Sync Pipe....190
Important Points about Access Control Filtering....190
To Configure the Sync Pipe to Filter Changes by Access Control Instructions....191
Contact Your Support Provider....191

Chapter 8: Configuring Synchronization with SCIM....193

About Synchronizing with a SCIM Sync Destination....194
Overview of SCIM Destination Configuration Objects....195
Tips for Syncing to a SCIM Destination....195
Renaming a SCIM Resource....196
Password Considerations with SCIM....196
Configuring Synchronization with SCIM....196
Configuring the External Servers....197
Configuring the Directory Server Sync Source....198
Configuring the SCIM Sync Destination....199
Configuring the Sync Pipe, Sync Classes, and Evaluation Order....199
Setting Up Communication with the Source Server(s)....201
Starting the Sync Pipe....201
Mapping LDAP Schema to SCIM Resource Schema....202
About the <resource> Element....202
About the <attribute> Element....203
About the <simple> Element....204
About the <complex> Element....204
About the <simpleMultiValued> Element....204
About the <complexMultiValued> Element....205

About the <subAttribute> Element....205
About the <canonicalValue> Element....205
About the <mapping> Element....206
About the <subMapping> Element....206
About the <LDAPSearch> Element....206
About the <resourceIDMapping> Element....207
About the <LDAPAdd> Element....207
About the <fixedAttribute> Element....207

Chapter 9: Managing Logging and Alerts....209

Working with Logs....210
Types of Log Publishers....210
Default Synchronization Server Logs....211
Viewing the List of Log Publishers....211
To View the List of Log Publishers....212
Sync Log Message Types....212
Creating New Log Publishers....213
To Create a New Log Publisher....213
To Create a Log Publisher Using dsconfig Interactive Command-Line Mode....214
Configuring Log Rotation....214
To Configure the Log Rotation Policy....215
Configuring Log Retention....215
To Configure the Log Retention Policy....215
Working with Administrative Alert Handlers....215
Configuring the JMX Connection Handler and Alert Handler....216
Configuring the SNMP Subagent Alert Handler....217
To Configure the SNMP Subagent Alert Handler....217
Running the Status Tool....217
To Run the Status Tool....221
To Search for a Specific Status Monitor....221
Monitoring the Synchronization Server....222
Monitoring Using SNMP....223
SNMP Implementation....223
Configuring SNMP....224
Configuring SNMP on AIX....226
MIBS....227

Chapter 10: Troubleshooting the Synchronization Server....229

About Synchronization Troubleshooting....230
About the Troubleshooting Tools....230
Troubleshooting Process Flow....231
Using the Sync Log....231
Sync Log Example 1....232
Sync Log Example 2....232
Sync Log Example 3....232
Troubeshooting Sync Failures....233
Troubleshooting "Entry Already Exists" Failures....233
Troubleshooting "No Match Found" Failures....235
Troubleshooting "Failed at Resource" Failures....236
Problems with the Metrics Engine Management Console: JVM Memory Issues....237
Working with the Collect Support Data Tool....238
Server Commands Used in the Collect Support Data Tool....238
JDK Commands Used in the Collect-Support-Data Tool....239

Linux Commands Used in the collect-support-data Tool..... 239
Solaris Commands Used in the collect-support-data Tool.....239
AIX Commands Used in the collect-support-data Tool.....240
MacOS Commands Used in the Collect Support Data Tool..... 241
Available Tool Options..... 241
To Run the Collect Support Data Tool.....242

Preface

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

Purpose of This Guide

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

Audience

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

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

Related Documentation

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

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

Document Conventions

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

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

Chapter 1 Introduction

The UnboundID® Synchronization Server is a point-to-point synchronization server between source and destination topologies comprised of the following:

- UnboundID® Directory Server
- UnboundID® Directory Proxy Server (3.x or later)
- Alcatel-Lucent® 8661 Directory Server
- Alcatel-Lucent® 8661 Directory Proxy Server (3.x or later)
- Oracle® Directory Server Enterprise Edition (DSEE 6.x, 7.x)
- Oracle® Directory Server (5.2 patch 3 or higher)
- Microsoft® Active Directory®
- Oracle® Database (10g, 11g)
- Microsoft® SQL Server (2005, 2008) systems

The Synchronization Server has a low cost of ownership with minimal administrative and hardware expenditures to provide a high performance synchronization solution. This chapter presents a general overview of the Synchronization Server:

Topics:

- *Overview of the Synchronization Server*
- *The Synchronization Problem*
- *The UnboundID Advantage*
- *Common Synchronization Use Cases*
- *Synchronization Server: How It Works*
- *Synchronization Modes of Operation*
- *Sync Operations*
- *Configuration Model*
- *Sync Control Flow Scenarios*
- *A Synchronization Example*
- *Available Tools Summary*
- *Summary*

Overview of the Synchronization Server

The UnboundID Synchronization Server is an efficient, pure Java-based server that provides high-throughput, low-latency, and bidirectional real-time synchronization between two endpoint topologies consisting of directory servers, directory proxy servers, and/or Relational Database Management Systems (RDBMS) systems. Designed to run on inexpensive hardware with little administrative maintenance (i.e., backups are not required), the UnboundID Synchronization Server provides an effective cost-per-performance solution for synchronizing data between LDAP-to-LDAP or LDAP-to-RDBMS directory topologies.

The Synchronization Server includes the following key features:

- High performance and availability with built-in redundancy to help ensure no downtime.
- Dataless virtual architecture for a small-memory footprint and easy maintenance.
- Hassle-free setup that allows you to transform and map attribute names, values, and DNs between endpoints. For directory server endpoints, this benefit allows you to make schema and directory information tree changes without the added costs of custom coding and scripting.
- Data flexibility and security, allowing you to replicate data and use advanced replication features in fractional, local data, filtered, or sub-tree replication scenarios.
- Multi-vendor directory server support including the UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Oracle/Sun Directory Server Enterprise Edition (DSEE 6.x, 7.x), Oracle/Sun Directory Server (5.2 patch 3 or higher), and Microsoft Active Directory.
- Relational Database Management System (RDBMS) support including Oracle Database (10g, 11g), and Microsoft SQL Server (2005, 2008) systems.
- Directory Proxy Server support including the UnboundID Directory Proxy Server (3.x) and the Alcatel-Lucent Directory Proxy Server (3.x).
- Notification support that allows real-time change notifications to be pushed to client applications or services as they occur.

The Synchronization Problem

Synchronization is the process of maintaining data consistency among applications, directories, and data sources in a networked environment. System administrators who use a directory as a writable user repository must ensure that the directory be exposed to all of its applications. However, exposing the user repository becomes problematic when consolidating applications and systems. Often the administrators find that many synchronization solutions lack features, such as writable partial replicas, or the ability to synchronize data with other multi-vendor directory servers and Relational Database Management System (RDBMS) systems.

Most companies employ a meta-directory or a virtual directory synchronization strategy as follows:

I. Meta-Directory (Datafull Approach). The meta-directory solution aggregates all data from the various sources and makes it available for its applications in a centralized directory. The centralized directory then can be updated with those changes pushed back out to the original data sources.

While the meta-directory approach may appear to be an easy-to-manage solution, there are some fundamental flaws that limit its ability for cost-effectiveness and performance:

- **Scalability Limitations**. To maintain a combined view of the data from its sources, the centralized meta-directory is often a bottleneck for any updates that must go through a single directory server instance when synchronizing from one endpoint to another.
- **Functionality Limitations**. The meta-directory solution must often integrate disparate company directories into a single distributed enterprise directory. Integrating data mismatches (e.g., schema variances, privilege differences, etc.) require additional solutions that limit ease-of-use.
- **Administrative and Hardware Cost Limitations**. Because a meta-directory stores a shadow copy of all of the source data that will be synchronized, it requires a large storage and memory footprint. This hardware requirement leads to additional hardware costs and increases the administrative burden of managing backups. The meta-directory solution also has difficulty in providing instantaneous failover between redundant instances.

II. Virtual Directory (Dataless Approach). Virtual Directories provide a consolidated view of the data without actually creating a physical centralized repository for directory information. When an application requests data from the virtual directory, the directory assembles the data and delivers it to the application in real time. However, to achieve synchronization between two backend directory topologies, virtual directories require that all applications update data through the virtual directory exclusively. This scenario prevents client applications from directly modifying the backend directory instances.

The UnboundID Advantage

Synchronization can be a challenging problem when integrating multiple data sources. UnboundID Corporation has a proven track record of successful deployments combined with many years of extensive synchronization experience to solve the problem.

The UnboundID Synchronization Server uses a dataless approach that synchronizes changes directly from the data sources in the background, so that applications can continue to update their data sources directly. The Synchronization Server does not store any data from the endpoints themselves, thereby reducing hardware and administration costs. The server's high-availability mechanisms also make it easy to fail over from the main synchronization server to its redundant instances.

Common Synchronization Use Cases

The UnboundID Synchronization Server synchronizes data across independent directory topologies using the following as source and destination endpoints:

- UnboundID Directory Server topologies
- UnboundID Directory Proxy Server (3.x) topologies
- Alcatel-Lucent 8661 Directory Server
- Alcatel-Lucent 8661 Directory Proxy Server (3.x) topologies
- Oracle Directory Server Enterprise Edition (DSEE 6.x, 7.x) topologies
- Oracle Directory Server (5.2 patch 3 or higher) topologies
- Microsoft Active Directory topologies
- Relational Database Management Systems (RDBMS) using Oracle (10g, 11g), Microsoft SQL Server (2005, 2008)

The typical deployment scenarios that require synchronization services involve synchronizing data during directory server migrations, replicating with advanced features during normal operations, synchronizing with Active Directory systems, performing real-time testing by obfuscating production data, synchronizing with database systems, and synchronizing through proxy servers.

Use Case: Synchronization during Directory Server Migrations

Directory Server migrations from one system to another can be complicated due to the mismatches in functionality from one system to another (for example, replication limitations, schema mismatches and others). Additionally, if problems arise during a migration, reverting the process can be especially difficult. The UnboundID Synchronization Server solves the migration/reversion problem by allowing you to leave the source deployment untouched, while a separate, synchronized topology of targeted servers is installed and tested. Modifications generated by an application in either topology are immediately synchronized to the other topology and are available to all applications. Once all of the applications have been tested against the new installation, the source directory servers can be phased out.

The general procedure for a migration (for example, from Sun Directory Server 5.x to UnboundID Directory Server) is as follows:

1. Leave the Oracle/Sun Directory Server 5.x in place.
2. Set up synchronization from the Oracle/Sun Directory Server 5.x to the UnboundID Directory Server and vice-versa.
3. Gradually migrate applications and data to the UnboundID Directory Server. You can use the UnboundID Directory Proxy Server in front of the Oracle/Sun Directory Server 5.x and the UnboundID Directory Server topologies to redirect some client applications to a particular topology.
4. In the event of a rare migration failure, the migration can be reverted back to the Oracle/Sun Directory Server 5.x if required.

Advanced Replication

The UnboundID Synchronization Server provides advanced features that extend the replication capability of its directory servers. The Synchronization Server can replicate parts of a Directory Information Tree (DIT) or subsets of entries using either fractional replication, local data replication, filtered replication, subtree replication, or a combination of these replication schemes. Traditionally, replication creates exact replicas of servers, including the same DIT structure, entries, and attributes. However, in many cases, replica servers need to store a subset of entries, a subset of attributes, or in some case extra attributes, compared to the full DIT in the primary master servers. Because all servers do not need full copies of the data, the extended replication features of the Synchronization Server improves the overall performance of the directory service and reduces hardware costs.

To provide an example scenario, a large telecommunications company is managing data replication between three divisions: billing, web, and network. The directory server for each division contains the same subscriber information. While the billing division is the authoritative source for the subscriber information, each division has its own unique directory structure. Each division replicates data between its own local servers using replication agreements that accommodate the division's unique DIT and schema. The Synchronization Server can address the needs of each division by synchronizing data across division boundaries using its advanced replication features.

Fractional Replication

Fractional replication is a form of partial replication that allows a subset of attributes to be replicated. By including only the data that are needed, this feature often reduces replication bandwidth. For example, if a replica only performs user authentications, then replication can be configured to only propagate the uid and userpassword attributes for a password policy, reducing the database size at the replica and the network traffic needed to keep the server in sync to this server. Furthermore, changes due to password policy attributes, such as account lockouts, can be replicated back to the main master servers. The UnboundID Synchronization Server supports fractional replication to any type of server.

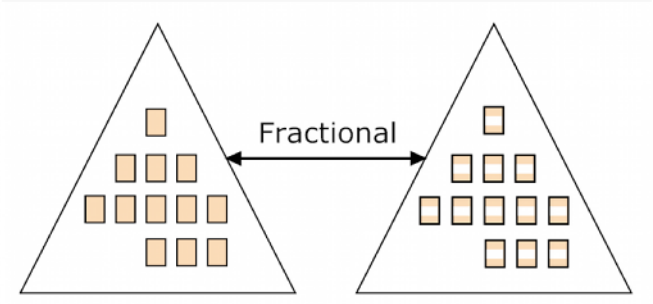


Figure 1: Fractional Replication

Returning again to the telecom company example, a subscriber can change their email address in a variety of ways:

- Calling the billing department

- Going to a retail store
- Logging on to the web site
- Using IVR on their telephone

No matter where the email address is changed, it needs to be reflected everywhere. If a subscriber changes his or her email address by calling the billing department, the Synchronization Server uses fractional replication to replicate only the updated email attribute of the subscriber's entry across the servers in other departments.

Local Data Replication

For fractional replication, application-specific repositories generally require directories that contain less data than their primary master servers. For local data replication, replicas tend to have more data than the primary master servers. For example, some applications need to store large amounts of data in a user entry, such as an XML blob of preference information, a sound file, or an image. Although the data could be required by only one application, the data itself can impact the server performance for all applications. The Synchronization Server can keep this local data isolated to only a few servers dedicated to this application without burdening the master corporate servers.

Returning to the large telecom company example, we can imagine that the web department uses a portal server and web applications that are not used by the other departments. The user preference information stored on the user entries in the web department's directory server is not replicated back to the other departments. Instead, this information is replicated only between the servers in the web department.

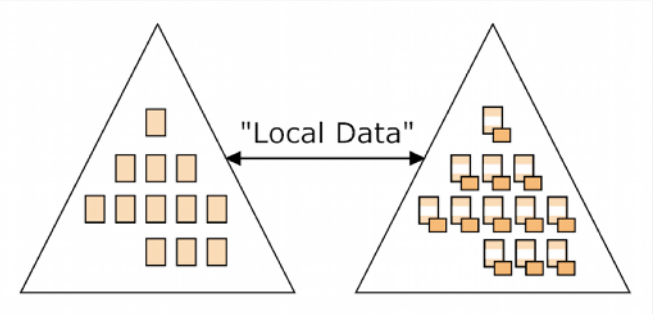


Figure 2: Local Data Replication

Filtered Replication

Filtered replication is a form of partial replication, where a replica contains only selected entries as determined by an LDAP filter. This feature allows directory instances to replicate only specific subtrees of a DIT, determined by base DNs returned using inclusion or exclusion filters. Applications that create an application-specific entry can be restricted to only those directories used by the application and not to every replica in the topology.

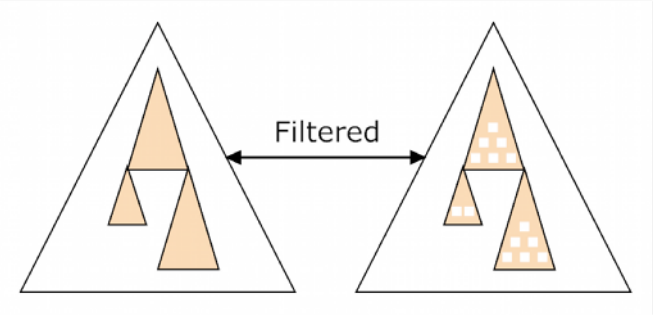


Figure 3: Filtered Replication

For example, the telecom example has an extranet directory server and a corporate directory server. The extranet directory server is used by traveling Sales staff to authenticate and use their email over the web. To access their email remotely, an employee must have a `web-access-enabled=true` flag set on their entry. Using filtered replication, the Synchronization Server can look for entries that have this flag set to true and replicate their changes back to the other corporate directories. If an employee changes their password in the extranet directory, this change will be replicated back to the master corporate directory.

Subtree Replication

In subtree replication, a replica contains only the selected entries as determined by a directory branch. This feature allows directory server instances to replicate only specific subtrees of a DIT, which are determined by inclusion or exclusion filtering on the base DNs.

For example, the large telecom company acquires a media company. The telecom company adds a subtree of data in its directory server for the media company, and the media company itself has its own on-premise LDAP directory server. Using a subtree replication protocol, data can be replicated between the media company's directory server and the main telecom directory server.

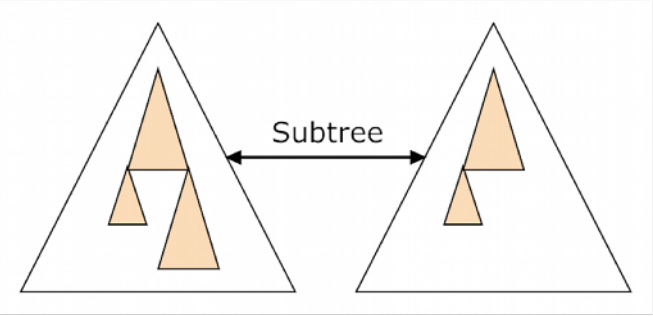


Figure 4: Subtree Replication

Advanced Replication Combinations

The UnboundID Synchronization Server can support various combinations of fractional and filtered replication for those applications that require it. For example, imagine that the telecom company operates in several countries in the European Union. The EU restricts the disclosure of anything considered personal data; rules inside individual countries can further restrict the type of data that can be transferred between country boundaries. While this data must remain in

the individual country's directory server, the corporate directory still needs to contain as much information as possible. Using fractional replication, only the parts of the entry that can legally cross country borders would be replicated back to the main corporate directory server.

Further, imagine that the main directory for the telecom company, dc=corp,dc=com contains subdirectories for each of the European directories, such as `ou=France`, `ou=Germany`, and `ou=Italy`. Locally, each country has its own unique directory data stored in its own DIT structure, such as `dc=corp-fr,dc=com` for France. Using subtree replication in combination with fractional replication, the Synchronization Server can replicate changes between a country's subtree in the master directory and each country's local directory, while adhering to local laws about the transfer of personal information.

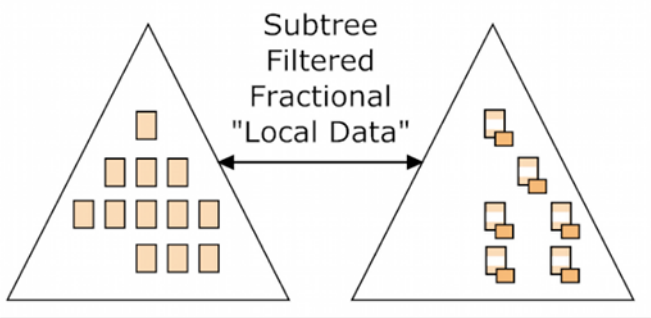


Figure 5: Advanced Replication Combinations

Use Case: Synchronization with Active Directory

The UnboundID Synchronization Server provides a Microsoft Active Directory mechanism that synchronizes ADD, MODIFY, and DELETE operations for user entries and individual attributes using Active Directory's DirSync control. For example, returning to the large tele-communications company example, the company uses multi-vendor directories including an Active Directory server in their respective data centers, some of which were acquired through acquisitions. Data must be successfully synchronized across these different directory servers in real-time, so that information can be up-to-date across these systems.

If real-time password synchronization is needed, the Synchronization Server also requires that a dedicated component, the UnboundID Password Sync Agent (PSA), be installed on all Active Directory domain controllers. The agent receives password changes from the Local Security Authority (LSA) and immediately hashes them with a secure 160-bit salted secure hash. The agent then sends the hashes to each UnboundID Synchronization Server instance in the topology over a secure LDAPS connection. If the synchronization server instance is down, the agent caches the change and retries synchronization until at least one of the servers has received the updates.

The agent is highly optimized with a small memory footprint. It securely handles sensitive data and uses a small, native DLL on the domain controller, which requires a single restart due to an Active Directory requirement. Subsequent updates to the DLL do not require a restart.

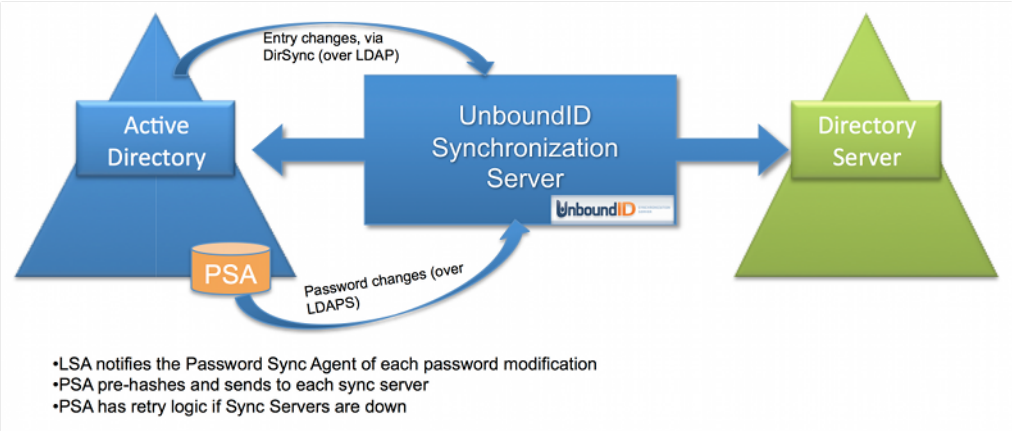


Figure 6: Active Directory Sync

Use Case: Synchronizing Realistic Test Environments

To secure sensitive user data, many companies have their own test environments that use synthetic data that may not be compatible with actual production data. This discrepancy can introduce problems when moving applications from a test environment to a production environment. The UnboundID Synchronization Server is capable of fully synchronizing test or stage servers with production servers while also obfuscating sensitive customer information, such as social security and phone numbers. The Synchronization Server can sync in real-time or on a nightly basis with little additional performance load on the production servers.

Use Case: Synchronizing with Relational Databases

In environments that store data on both directory servers and databases (Oracle 11g, 10g and Microsoft SQL Server 2005, 2008), the UnboundID Synchronization Server can synchronize data on both systems. This solution is more flexible than having to sync through a virtual directory. You can configure the UnboundID Synchronization Server to establish the directory server or the database as the authoritative data source.

Use Case: Synchronizing through Proxy Servers

Many data centers use proxy servers in front of a backend set of directory servers in load-balanced and/or entry-balancing deployments. The UnboundID® Synchronization Server provides the ability to synchronize data through such proxy deployments. For example, the large telecommunications company deploys proxy servers in an entry-balancing deployment to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance. The proxy server fronts a backend set of directory servers at one data center, while another proxy server at a different data center fronts another set of directory servers. The Synchronization Server can successfully synchronize data across these two topologies.

Synchronization Server: How It Works

The UnboundID Synchronization Server is a lightweight, standalone, 100% Java solution that provides low-latency, highly-available, point-to-point synchronization. The Synchronization Server shares many of the reliable components with the UnboundID Directory Server for easy installation and configuration.

Point-to-Point Bidirectional Synchronization

The UnboundID Synchronization Server performs point-to-point synchronization between a source endpoint and a destination endpoint. An endpoint is defined as any Source or Destination topology of directory or database servers comprised of any combination of either UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Sun Directory Server Enterprise Edition (DSEE 6.x, 7.x), Sun Directory Server (5.2 patch 3 or higher), Microsoft Active Directory, Oracle Database (10g, 11g), and Microsoft SQL Server (2005, 2008) systems.

The Synchronization Server provides the ability to sync data in one direction or bidirectionally between endpoints. For example, in a migration phase from Sun Directory Server to UnboundID Directory Server, you can sync in one direction from the source server to a QA stage server for testing purposes. For one-way synchronization, the source server is the authoritative endpoint as it generates all of the changes in the system. Bidirectional synchronization allows for parallel active installations between the source and the destination endpoints. In a bidirectional synchronization configuration, both sets of endpoints (i.e., the source and the destination) are authoritative for the same set of attributes or for different sets of data.

The Synchronization Server also contains no single point of failure, either for detecting changes or for applying changes. The Synchronization Server instances themselves are redun- dant, so that you can have multiple synchronization server instances running at a time, but only the server with the highest priority is actively syncing changes. Further, the stand-by servers are constantly polling the active server instance to update their persistent state. This state contains the minimum amount of information needed to begin synchronization where the primary server left off, which logically is the last processed change number for the source server. In the case of a network partition, multiple synchronization servers can synchronize simultaneously without causing problems as they each verify the full entry before making any changes.

Figure 7 shows a typical UnboundID Synchronization Server deployment. The Synchronization Server looks for any changes on the main source server on the left and applies those changes to the destination server on the right. In the diagram, the bold lines indicate the primary (active) connections within the synchronization network that show the directional path of the changes. The synchronization servers will communicate with these servers if they are up. The gray lines are possible failover connections if any component is down.

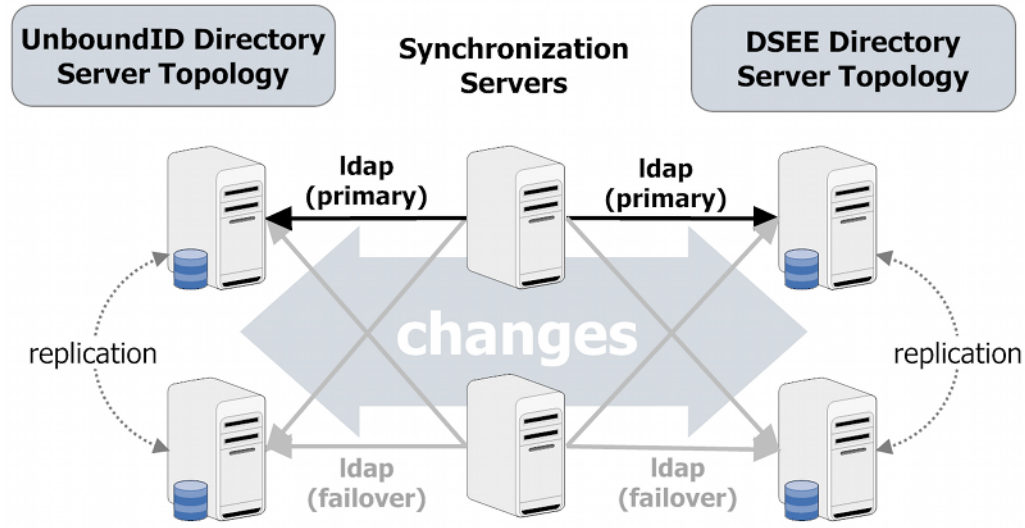


Figure 7: Synchronization Topology

Synchronization Architecture

The UnboundID Synchronization Server uses a virtualized, dataless approach and never permanently stores any directory data locally. This dataless approach eliminates the need for backups and additional hardware components, such as large disk installations. The log files, administrator entries, configuration, sync state information are stored as flat files (LDIF format) within the system. No additional database is required.

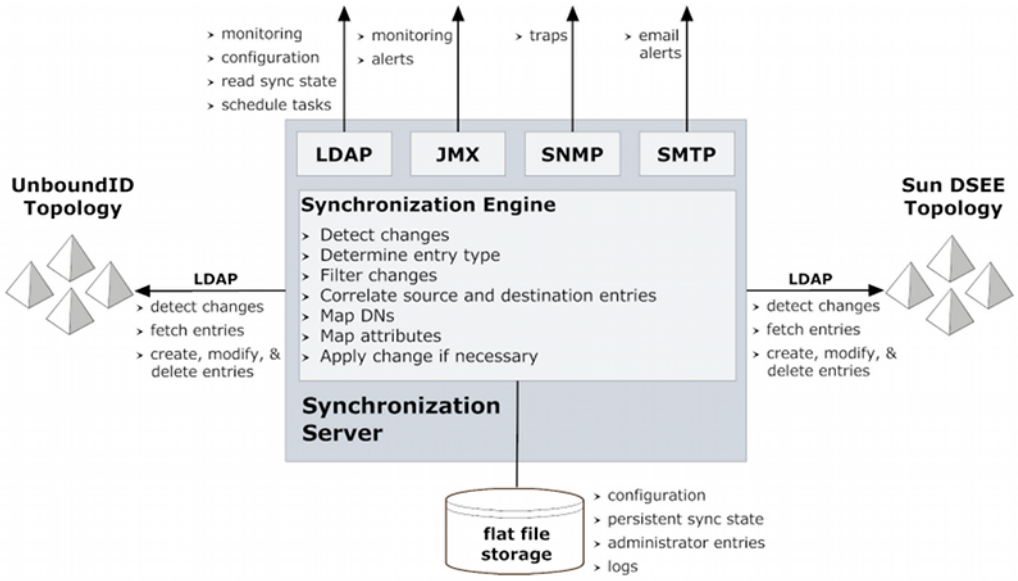


Figure 8: Synchronization Architecture

Change Tracking

To track changes in each endpoint system, the UnboundID Synchronization Server uses the change log mechanism that is most efficient for each platform.

- For UnboundID Directory Server or Alcatel-Lucent 8661 Directory Server topologies, the UnboundID Synchronization Server uses the servers's LDAP Change Log for modification detection.
- For Sun DSEE or Oracle/Sun Directory Server topologies, the UnboundID Synchronization Server uses the server's Retro Change Log, which provides a detailed summary of each change applied to the directory.
- For Active Directory, the UnboundID Synchronization Server uses the DirSync control, which polls for object attribute changes.
- For RDBMS systems, the UnboundID Synchronization Server uses an UnboundID Server SDK plug-in to interface with a customized RDBMS change log table. Database triggers on each table record all INSERT, UPDATE, and DELETE operations to the change log table.

Each directory instance stores a separate entry under `cn=changelog` for every modification made to the directory. The UnboundID Synchronization Server provides full control over the synchronization process by determining which entries are synchronized, how they are correlated to the entries at the destination endpoint, and how they are transformed into the destination schema.

Monitoring and Alerts

The Synchronization Server supports several industry-standard, administrative protocols for monitoring and alerts:

- LDAP. Used for monitoring, configuration, server state, tasks.
- JMX. Used for monitoring and alerts.
- SMTP. Used for email alerts.

The UnboundID Synchronization Server provides an administrative alert framework that can be used to notify administrators of any significant warnings, errors, or other noteworthy events that occur in the server. Existing alert handlers can notify administrators through log messages, email, or JMX notifications. All administrative alerts are also exposed over LDAP as entries below a base DN `cn=alerts`. You can use the persistent search operation to ensure that you are automatically notified over LDAP of any new alerts generated by the server. The administrator can select the admin action for each type of alert based on the severity level or the specific type of alert. For example, it may be desirable to log information about all types of alerts, but only generate email messages. Typical alert events are startup/shutdown, applied configuration changes, or synchronized resources unavailable.

Logging

The UnboundID Synchronization Server provides standard logs (`sync`, `access`, `error`, `failed-operations`, `config-audit.log`, `debug`) to troubleshoot any issues. The server can also be configured for multiple active sync logs. For example, you can log each detected change, each dropped change, each applied change, or each failed change.

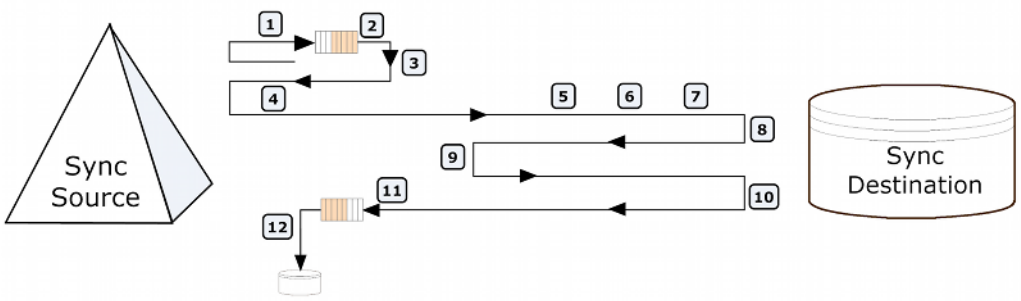
Synchronization Modes of Operation

The UnboundID Synchronization Server runs as a standalone Java process with two complementary synchronization modes of operation: standard and notification mode.

Standard Synchronization Mode

In standard synchronization mode, the Synchronization Server polls the directory server change log for creates, modifies, and delete operations on any entry. The Synchronization Server fetches the full entries from both the source and destination endpoints, and compares them to produce the minimal set of changes to bring the destination in sync with the source.

Figure 9 shows the standard synchronization change flow that syncs data between two end-point servers. The source or destination endpoint can be a directory server, a directory proxy server, or a database server. Although not pictured, the changes are processed in parallel, which increases throughput and offsets network latency.



1 Dedicated thread periodically polls changelog for new changes & enqueues them as SyncOps

2 Change application thread dequeues SyncOp

3 Grab write lock for entry and read lock for its parent

4 Fetch latest copy of entry from source

5 Determine Sync Class using base DNs and filters

6 Ensure operation should be synchronized

7 Map attributes and DN

8 Fetch destination entry

9 Compute minimal changes to bring entry into sync

10 Apply change

11 Release locks and acknowledge change so that it is not detected again

12 Periodically persist changelog state

Figure 9: Standard Synchronization Mode Change Flow

Notification Synchronization Mode

In notification synchronization mode, the Synchronization Server skips the fetch and compare phases of processing and simply notifies the destination that a change has happened and provides it with the details of the change. Notification mode is currently available for the UnboundID and Alcatel-Lucent 8661 brands of directory and proxy servers only.

For more information on notificaiton mode, see *Configuring Notification Mode*.

Sync Operations

The UnboundID Synchronization Server provides seamless integration between disparate systems to transform data using attribute and DN mappings. To validate that the mappings are correct, administrators can run a bulk resynchronization operation to test the synchronization settings. Once the topology has been verified to work as planned, administrators can start real-time synchronization globally or on specific sync pipes.

Data Transformations

Data transformations alter the contents of synchronized entries between the source and destination directory server topologies to transparently handle variances in attribute names, attribute values, or DN structures. When the UnboundID Synchronization Server synchronizes entries between a source and a destination server, it can be configured to change the contents of these entries using attribute and DN mappings, so that neither server needs be aware of the transformations.

- **Attribute Mapping**. The Synchronization Server can transparently rename any attribute in the entry to fit the schema definitions from the source endpoint to the destination endpoint. This mapping makes it possible to synchronize information stored in one attribute in one directory server topology to an attribute with a different name in another directory server topology, or to construct an attribute using portions of the source attribute values.
- **DN Mapping**. The Synchronization Server can transparently alter any DNs referenced in the entries. This mapping makes it possible to synchronize data from a topology that uses one DIT structure to a system that uses a different DIT structure.

Resync

In resync operations, the Synchronization Server performs a bulk comparison of entries on source topologies and destination topologies. It streams all entries (or those entries matching certain criteria) from the source, and either updates the corresponding destination entries or reports those that are out-of-sync. Administrators run a resync operation using the resync utility in the bin folder (UNIX or LINUX) or bat folder (Windows).

Resync is used for any of the following tasks:

- Verify that the two endpoints are in-sync after an initial configuration.
- Initially populate a newly configured target endpoint.
- Validate that the server is behaving as expected. The resync tool has a `--dry-run` option that validates that sync is operating properly without updating any entries. This option also can be used to check attribute or DN mappings.
- Perform scheduled (for example, nightly) synchronization in place of real-time sync.
- Recover from a failover by resyncing entries that were modified since the last backup was taken.

Resync also allows for fine control over what can be synchronized. You can control the following items:

- Include or exclude any source and destination attributes.
- Apply an LDAP filter to only sync entries created since that last time you ran the tool (for example, `createTimeStamp >= 2011333200-0600`).
- Synchronize only creates or only modifications.
- Change the logging verbosity.
- Set a limit on how fast the resync runs (for example, only 2000 operations/second) to limit the impact on endpoint servers.

Real-Time Synchronization

In real-time operations, the Synchronization Server polls the source server for changes and synchronizes the destination entries immediately. Once the Synchronization server determines that a detected change should be synced, it fetches the full entry from the source. Then it searches for the corresponding entry in the destination endpoint using flexible correlation rules and applies the minimum set of changes to bring the attributes that were modified into sync. The server fetches and compares the full entries to make sure it does not synchronize any old data from the change log.

Administrators run real-time synchronization using the `realtime-sync` utility in the bin folder (UNIX or LINUX) or bat folder (Windows). In most applications, after you configure a synchronization topology, run resync to get the endpoints in-sync, and then run `realtime-sync` to start global synchronization.

Realtime-sync is used for any of the following tasks:

- Start synchronization globally or for specific sync pipes only.
- Stop synchronization globally or for specific sync pipes only.
- Set a start point at which synchronization should begin syncing changes at the beginning or end of the change log, at a specified change number, at a specified change sequence number, or at a specified time frame that rewinds back to a certain point in the change log.

About the Sync Retry Mechanism

In both standard and notification mode, the Synchronization Server is designed to quickly synchronize data between two endpoints and attempt a retry should an operation fail for any reason. The retry mechanism involves two possible retry levels, which are configurable on the Sync Pipe configuration using advanced Sync Pipe properties. (For detailed information, see the UnboundID Synchronization Server Reference Guide for the Sync Pipe Configuration parameters.)

To summarize, retry involves two possible levels:

1. **First Level Retry**. If an operation fails to synchronize for any reason, the Synchronization Server will attempt a configurable number of retries immediately (with some backoff, i.e., delay). The total number of retry attempts is set by the value set in the `max-operation-`

`attempts` property on the Sync Pipe. This property applies only to error that a limited amount of retries. The property indicates how many times a worker thread should retry the operation before putting the operation into the second level of retry called the delayed-retry queue or failing the operation altogether.

Note: The following additional advanced Sync Pipe properties are present should you require more fine-tuning:

- **retry-backoff-initial-wait**. Specifies the initial amount of time to wait before retrying an operation for the first time. Default is 100 milliseconds.
- **retry-backoff-max-wait**. Specifies the maximum amount of time to wait between operation retry attempts. Default is 10 seconds.

One of the following two properties can be used to increase the amount of time, either explicitly given in milliseconds or by a percentage between consecutive retry attempts.

- **retry-backoff-increase-by**. Specifies the specific amount of time to increase the backoff in between consecutive retry attempts. Default is 0 seconds.
- **retry-backoff-percentage-increase**. Specifies the percentage of time to increase the backoff in between consecutive retry attempts. Default is 100 percent, which will double the amount of time between each consecutive retry attempt.

For more detailed information, see the *UnboundID Synchronization Server Reference Guide*.

---

2. **Second Level Retry**. Once the `max-operation-attempts` property has been exceeded, the retry is sent to the second level retry phrase called the delayed-retry queue. The delayed-retry queue uses two advanced Sync Pipe properties to determine the number of times a failed operation should be retried in the background after a specified delay.

   Operations that make it to this level will be retried after the `failed-op-background-retry-delay` property (default: 1 minute). Next, the Synchronization Server checks the `max-failed-op-background-retries` property to determine the number of times a failed operation should be retried in the background. By default, this property is set to 0, which indicates that no background retry should be attempted in the background, and that the operation should be logged as a failed operation. However, if you set this property to a non-zero value, the operation will be retried in the background up to that number of times. Having operations retried in the background can hold up processing other changes since the Synchronization Server will only process up to the next 5000 changes while waiting for a retried operation to complete (one way or the other).

Note: Retry can be controlled by the custom endpoint based on the type of exception that is thrown on an error. When throwing an exception, the endpoint code can signal that a change should be 1) aborted, 2) retried a

limited number of times, or 3) retried an unlimited number of times. Some error, such as endpoint server down, should be retried indefinitely.

3. If the `max-failed-op-background-retries` property has been exceeded, then the retry is logged as a failure and appears in the sync and the `sync-failed-ops` logs. Note that the sync log records the results of all operations, while the `sync-failed-ops` log records only failed operations.

Configuration Model

The UnboundID Synchronization Server supports a flexible configuration model that is designed for easy installation and maintenance based on a comprehensive set of command-line tools and a graphical console. The Synchronization Server supports a defined set of configuration parameters that determine how synchronization takes place between directories or databases. The server can be configured remotely or locally with all configuration changes taking effect on-the-fly.

The configuration parameters are listed in Table 1 and the configuration model is summa- rized in Figure 10.

Table 1: UnboundID Synchronization Server Configuration Components

Component	Description
Sync Pipe	Defines a single synchronization path between the source and destination topologies. Every Sync Pipe has one or more Sync Classes that controls how and what is synchronized. Multiple Sync Pipes can run in a single UnboundID Synchronization Server instance.
Sync Source	Defines the directory topology that is the source of the data to be synchronized. When data in the Sync Source changes, it is synchronized to the Sync Destination topology. A Sync Source can reference one or more external servers of the appropriate type (UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Oracle/Sun DSEE, Oracle/Sun Directory Server 5.x, Microsoft Active Directory, Oracle, Microsoft SQL Server).
Sync Destination	Defines the topology of directory servers where changes detected at the Sync Source are applied. A Sync Destination can reference one or more external servers of the appropriate type (UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Oracle/Sun DSEE, Oracle/Sun Directory Server 5.x, Microsoft Active Directory, Oracle, Microsoft SQL Server).
External Server	Defines a single server in a topology of identical, replicated servers to be synchronized. For an LDAP server, you must define the host, port, SSL, bind DN, and bind password. A single external server configuration object can be referenced by multiple Sync Sources and Sync Destinations.
Sync Class	Defines the operation types (e.g., creates, modifies, or deletes) and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. A source entry is in at most one Sync Class and is determined by a base DN and LDAP filters. A Sync Class can reference zero or more Attribute Maps and DN Maps, respectively. Within a Sync Pipe, a Sync Class is defined for each type of entry that needs to be treated differently. For example, entries that define attribute mappings or entries that should not be synchronized at all. A Sync Pipe must have at least one Sync Class but can refer to multiple Sync Class objects.
DN Map	Defines mappings for use when destination DNs differ from source DNs. These mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can be used only once. The wild card values can be used in the `to-dn-pattern` attribute using {1} and their original index position in the pattern, or {attr} to match an attribute value. For example: `**,dc=myexample,dc=com->{1},o=example`

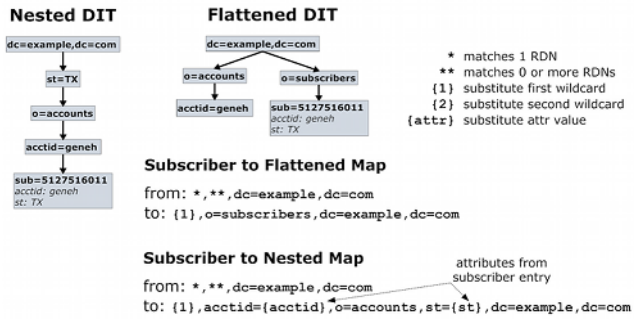
Component	Description
	You can also use regular expressions and attributes from the user entry. For example, the following mapping constructs a value for the uid attribute, which is the RDN, out of the initials (first letter of `givenname` and `sn`) and the employee ID (the `eid` attribute). `uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example` The following diagram shows how a nested DIT can be mapped to a flattened structure:  A Sync Class can reference zero or more DN maps. Multiple Sync Classes can share the same DN Map.
Attribute Map & Attribute Mappings	Defines a mapping for use when destination attributes differ from source attributes. An Attribute Map is a collection of Attribute Mappings. There are three types of Attribute mappings: • **Direct Mapping**. Attributes are directly mapped to another attribute: For example, employeenumber->employeeid • **Constructed Mapping**. Destination attribute values are derived from source attribute values and static text. For example: `{givenname}.{sn}@example.com->mail` • **DN Mapping**. Attributes are mapped for attributes that store DNs. You can reference the same DN mappings that map entry DNs. For example, you could have a manager attribute that has the value `uid=jdoe,ou=People,dc=example,dc=com`. A Sync Class can reference multiple Attribute Maps. Multiple Sync Classes can share the same Attribute Map.

Figure 1-10 shows a summary of the configuration model and how the configuration objects can be referenced.

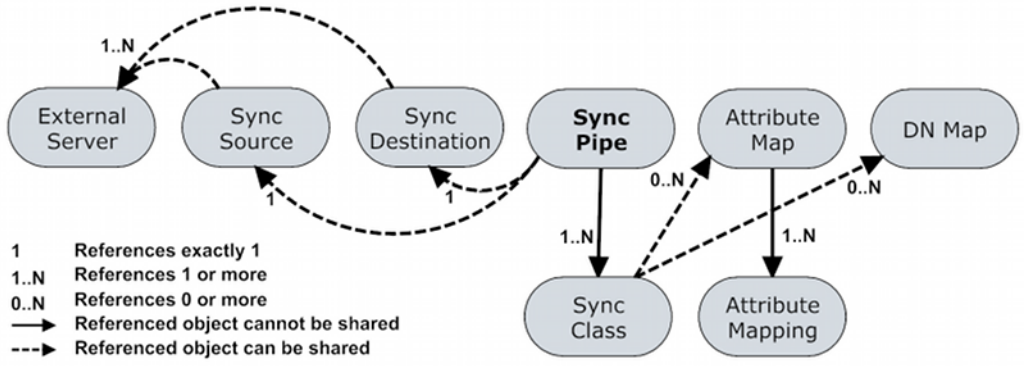


Figure 10: Configuration Model Referenced Objects

Sync Control Flow Scenarios

The Synchronization Server processes changes by fetching the most up-to-date, full entries from both sides and then compares them. This process flow is called standard synchronization mode. The processing flow differs depending on the type of Synchronization Server change (ADD, MODIFY, DELETE, MODDN) that is requested. Table 1-2 to 1-6 shows the control flow diagrams for the sync operations, especially for those cases when a MODIFY or a DELETE operation is dropped. The sync log records all completed and failed operations.

Table 1-7 shows the control flow for notification synchronization mode, which does not use the fetch-and-compare processes used in standard mode. For more information, see *Configuring Notification Mode* on page 174.

Table 2: Standard Modify

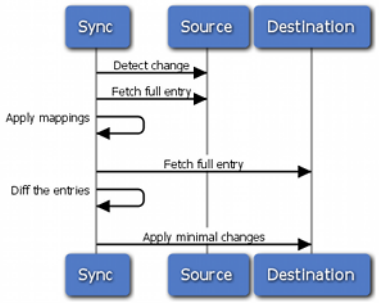
1. Detect change from the change log table on the source. **2.** Fetch the entry or table rows from affected tables on the source. **3.** Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row. **4.** Fetch the entry or table rows from affected tables on the destination. **5.** Diff the computed destination entry and actual destination entry. **6.** Apply the minimal set of changes to the destination to bring it in sync.	

Table 3: Standard Add

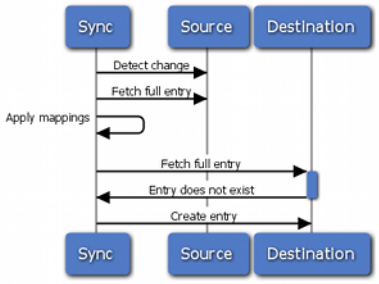
1. Detect change from the change log table on the source. **2.** Fetch the entry or table rows from affected tables on the source. **3.** Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row. **4.** Fetch the entry or table rows from affected tables on the destination. **5.** The entry or table row does not exist on the destination. **6.** Create the entry or table row.	

Table 4: Standard Delete

1. Detect delete from the change log table on the source. **2.** Fetch the entry or table rows from affected tables on the source. **3.** Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row. **4.** Fetch the entry or table rows from affected tables on the destination. **5.** The entry or table row exists on the destination.	

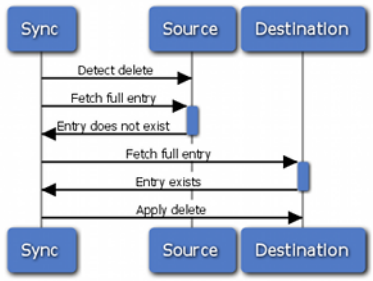
6. Apply the delete on the destination.	

Table 5: Standard Delete After Source Entry Was Re-Added

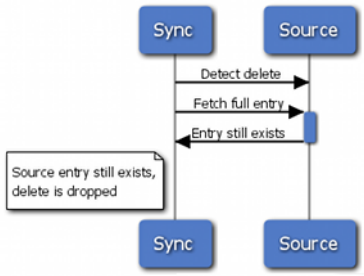
1. Detect delete from the change log table on the source. **2.** Fetch the entry or table rows from affected tables on the source. **3.** The entry or table row exists on the source. **4.** Delete request is dropped.	

Table 6: Standard Modify After Source Entry is Deleted

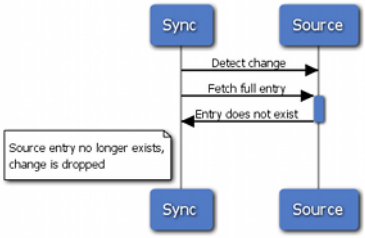
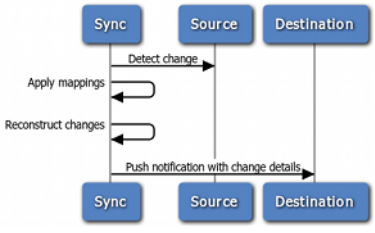
1. Detect change from the change log table on the source. **2.** Fetch the entry or table rows from affected tables on the source. **3.** The entry does not exist. **4.** Change request is dropped as the source entry no longer exists.	

Table 7: Notification Add, Modify, ModifyDN and Delete

1. Detect change from the change log table on the source. **2.** Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row. **3.** Reconstruct changed entries. **4.** Push notification with change details to the destination.	

A Synchronization Example

Figure 11 shows a synchronization migration example from a Sun Directory Server Enterprise Edition (DSEE) topology to the UnboundID Directory Server topology with a change in the DIT structure to a flattened directory structure. The Sync Pipe connects the Sun Directory Server topology as the Sync Source and the UnboundID Directory Server topology as the Sync Destination. Each endpoint is defined with three external servers in their respective topology. The sync pipe destination has its base DN set to o=example, which is used when performing LDAP searches for entries.

Two sync classes are defined: one for Subscribers, and one for Accounts. Each Sync Class uses a single "Sun DS to UnboundID Attribute Map" that has four attribute mappings defined. All other attributes are mapped as is.

Each sync class also defines its own DN Maps. For example, the Accounts Sync Class uses a DN Map, called UnboundID Account Map, that is used to flatten a hierarchical DIT, so that the Account entries appear directly under ou=accounts. The DN Map is as follows:

```
*,**,o=example -> {1},ou=accounts,o=example
```

With this mapping, if an entry DN has uid=jsmith,ou=people,o=example, then "*" matches uid=jsmith, "**" matches ou=people, and {1} matches `uid=jsmith`. Thus, `uid=jsmith,ou=people,o=example` gets mapped to `uid=jsmith,ou=accounts,o=example`. A similar map is configured for the Subscribers Sync Class.

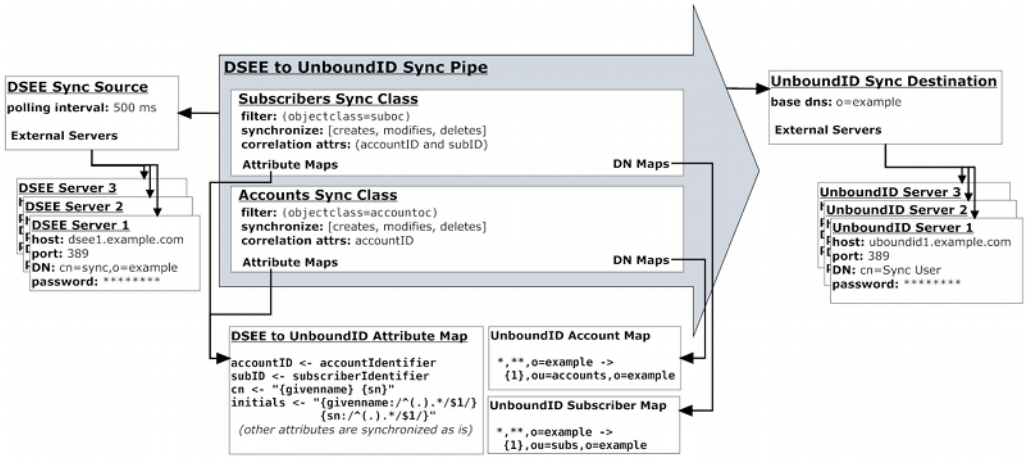


Figure 11: A Typical Synchronization Topology Configuration

Available Tools Summary

The UnboundID Synchronization Server supports a flexible configuration framework that stores its settings in a flat file for a small memory footprint and easy access. Administrators can access the configuration using the UnboundID® Sync Management Console or using the full suite of command-line tools in the bin directory for UNIX® or Linux® systems and the bat directory for

Microsoft® Windows® systems. For detailed information and examples of the command-line utilities, see the *UnboundID Synchronization Server Command Line Tool Reference*.

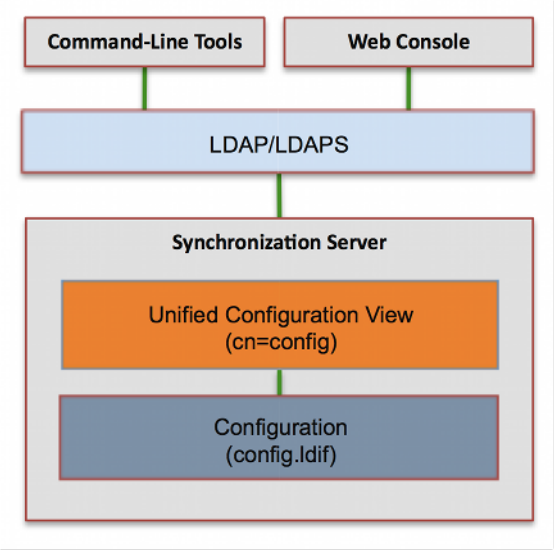


Figure 12: UnboundID Synchronization Server Configuration Framework

You can view the Help information for each command-line tool by typing `--help` with the command (or type the short forms, -? and -H).

```
$ bin/resync -H
```

For those utilities that support additional subcommands (`dsconfig`, `dsframework`), you can also get more detailed subcommand information by typing `--help` with the subcommand:

```
$ bin/dsconfig list-log-publishers --help
```

Summary

The UnboundID Synchronization Server provides an excellent "total cost versus high performance" solution for your synchronization requirements. The server provides point-to-point, bidirectional synchronization with immediate failover and is ideal for large production environments that depend on highly available performance. This parallel, high-throughput, 100% Java solution requires little administrative cost and maintenance. UnboundID has leveraged its years of directory and synchronization expertise in the Identity Management industry to provide the ideal solution to meet your data center and enterprise synchronization needs.

Chapter 2 Installing the Synchronization Server

You can begin the setup process using one of the UnboundID Synchronization Server's easy-to-use installation modes. Then, you can configure the Synchronization Server using the `create-sync-pipe-config` tool, the `dsconfig` command-line tool, or the UnboundID Sync Management Console if you prefer a graphical interface. Other instructions are provided to install redundant failover servers.

This chapter presents the various installation options and procedures available to administrators:

Topics:

- *Before You Begin*
- *About the Server Installation Modes*
- *Installing the UnboundID Synchronization Server in Interactive Mode*
- *Installing the UnboundID Synchronization Server in Non-Interactive Mode*
- *Installing the Synchronization Server with a Truststore in Non-Interactive Mode*
- *Running the Server*
- *Stopping the Directory Server*
- *About the Server Uninstallation Modes*
- *Installing the Sync Management Console*
- *Updating the Synchronization Server*
- *Installing a Redundant Failover Server*
- *Removing a Redundant Server*

Before You Begin

To begin the installation process, obtain the latest zip release bundle for the Synchronization Server and the Sync Management Console from UnboundID and unpack them in a folder of your choice.

Supported Operating Platforms

Multi-Platform Support. The UnboundID® Synchronization Server is a pure Java application. It is intended to run within the Java Virtual Machine on any Java 6 Standard Edition (SE) or Enterprise Edition (EE) certified platform including, but not limited to the following:

- RedHat® Enterprise Linux® 5.5 (natively or running under VMWare® ESX 4.0 Update 2)
- Oracle® Solaris™ 10 update 9
- RedHat® CentOS 5.5 and 6.2
- AIX® 6.1

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.

Software Requirements: Java 6

For optimized performance, the UnboundID® Synchronization Server requires Java 6 (minimum required Sun/Oracle JDK 1.6.0_26, recommended Sun/Oracle JDK 1.6.0_31 or JRE 1.6.0 IBM J9 2.4) for 32-bit and 64-bit architectures, respectively, depending on your system requirements.

Even if your system already has Java installed, you may want to create a separate Java 6 installation for use by the UnboundID® Synchronization Server to ensure that updates to the system-wide Java installation do not inadvertently impact the directory server. This setup requires that the JDK, rather than the JRE, for both the 32-bit and 64-bit versions or both depending on your operating system, be downloaded.

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

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

About the Server Installation Modes

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

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

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

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

Installing the UnboundID Synchronization Server in Interactive Mode

The `setup` tool provides an interactive text-based interface to install a Synchronization Server instance. You can install the Synchronization Server by entering the required input as presented by the prompts.

To Install the Synchronization Server in Interactive Mode

1. Change to the server root directory.

```
$ cd UnboundID-Sync
```

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

```
$ ./setup
```

Note: If your *JAVA_HOME* environment variable is set to an older version of Java, you must explicitly specify the path to the Java 6 JDK installation during setup. You can either set the *JAVA_HOME* environment variable with the Java 6 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. If you are adding this server to an existing Synchronization Server topology, type `yes`. Otherwise, press **Enter** to accept the default (`no`).

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

```
What would you like to use as the initial root user DN for the Synchronization Server
 [cn=Directory Manager]:
Please provide the password to use for the initial root user:
Please re-enter the password for confirmation:
```

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

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

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

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

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

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

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

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

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

```
By default the server listens on all available network
interfaces for client connections. Would you like to specify
particular addresses on which this server will listen for
```

```
client connections? (yes / no)(no):
```

10. 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 Synchronization Server is the primary application and no other processes consume a significant amount of memory.

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

11. Enter the maximum amount of memory you want the Synchronization Server to allocate to the server and tool. In this example, the maximum allowed for the server is 1 gigabytes.

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

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

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

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

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

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

Start Server when the configuration is completed

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

Enter choice [1]:

Configuring Synchronization Server..... Done.
Starting Synchronization Server....... Done.

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

14. Register the server to a server group. See *Configuring Server Groups* on page 72 for instructions.

15. At this point, you have the following options depending on your specific configuration:
    a) Build a configuration if the command-line wizard was not employed.
    b) Determine if a bulk import or resync is necessary.
    c) Determine if a `realtime-sync set-startpoint` is necessary.
    d) Enable syncing with `realtime-sync start`.

Installing the UnboundID Synchronization Server in Non-Interactive Mode

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

To Install the Synchronization Server in Non-Interactive Mode

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

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

Installing the Synchronization Server with a Truststore in Non-Interactive Mode

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

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

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

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

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

```
  --useJavaTrustStore /path/to/devtruststore.jks \
  --acceptLicense

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

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

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

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

Running the Server

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

To Start the Synchronization Server

Use bin/start-sync-server to start the server.

```
$ bin/start-sync-server
```

To Start the Synchronization Server with Global Sync Disabled

When restarting the Synchronization Server, you may want to start the server but not have synchronization begin right away.You can run the `start-sync-server` command with the `--globalSyncDisabled` option to start the server without synchronization. Note that this option does not modify the configuration, you must run the `realtime-sync` tool to restart the global synchronization.

To Run the Server as a Foreground Process

1. Enter bin/start-sync-server with the `--nodetach` option to launch the Synchronization Server as a foreground process.

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

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

To Start the Server at Boot Time

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

1. Create the startup script.

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

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

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

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

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

Stopping the Directory Server

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

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

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

To Stop the Server

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

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

To Schedule a Server Shutdown

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

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

  ```
  $ bin/stop-ds --stopTime 201206061345OOZ --port 1389 \
    --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
    --stopReason "Scheduled offline maintenance"
  ```

To Restart the Server

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

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

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

To Restart the Synchronization server using an Internal Restart

To avoid destroying and re-creating the JVM, use an internal restart, which can be issued over LDAP. The internal restart will keep the same Java process and avoid any changes to the JVM options.

- Go the server-root directory. Using a loop back interface, run the `stop-sync-server` command with the `-R` or `--restart` options.

  ```
  $ bin/stop-sync-server --restart --hostname 127.0.0.1 --port 389 \
    --bindDN "cn=Directory Manager" --bindPassword secret
  ```

About the Server Uninstallation Modes

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

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

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

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

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

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 Synchronization Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Synchronization Server.

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

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

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

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

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

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

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

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

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

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

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

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

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

Enter choice [3]:
```

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

To Uninstall the Server in Non-Interactive Mode

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

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

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

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

To Uninstall Selected Components in Non-Interactive Mode

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

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

Installing the Sync Management Console

The UnboundID® Synchronization Server provides a graphical web application tool, the UnboundID Sync Management Console. The Sync Management Console provides configuration and schema management functionality in addition to monitoring and server information. The Sync Management Console is a web application that must be deployed in a servlet container that supports the servlet API 2.5 or later. The installation using Apache Tomcat is described below for illustration purposes only.

Note: Like the `dsconfig` tool, all changes made using the Sync 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.

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

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

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

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

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

```
$ cd apache-tomcat-<version>/webapps/Sync console
$ jar xvf syncconsole.war
```

6. Optional. Edit the `WEB-INF/web.xml` file to point to the correct directory instance. Change the host and port to match your directory. 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 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:

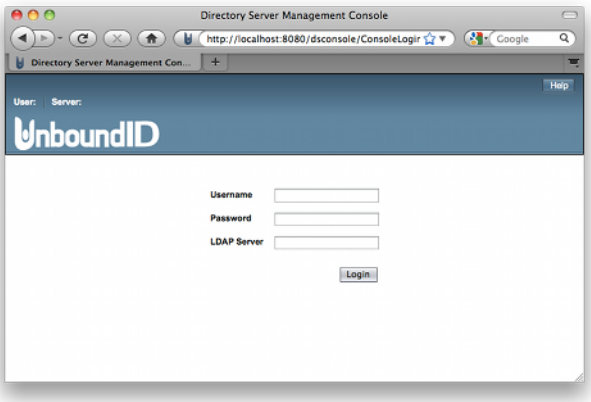
```
<session-config>
     <session-timeout>120</session-timeout>
</session-config>
```

   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 directory proxy server is restarted, so consider changing the value before starting the server for the first time.

8. Start the Synchronization Server if it is not already running, and then start the Sync 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/Sync console`. By default, Tomcat listens on port 8080 for HTTP requests.



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

To Log into the Sync Management Console

1. Go to the installation directory.

```
$ cd UnboundID-Sync
```

2. Start the Synchronization Server.

```
$ bin/start-sync-server
```

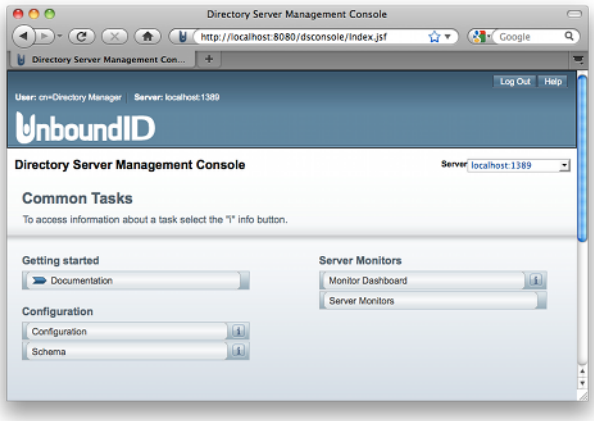
3. Start the Apache Tomcat application server.

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

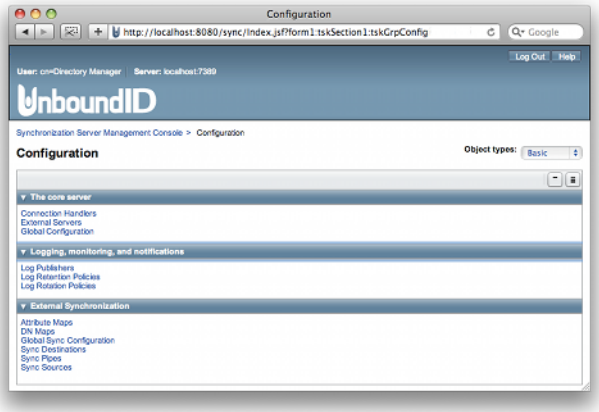
4. Open a browser to http://hostname:8080/Sync console/.

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

6. On the Sync 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 Uninstall the Sync Management Console

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

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

2. Remove the `webapps/Sync console` directory.

   ```
   $ rm -rf webapps/Sync console
   ```

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.

To Upgrade the Sync Management Console

1. Shut down the console and servlet container.

2. In the current deployment of the Sync Management Console, move the `webapps/Sync console/WEB-INF/web.xml` file to another location.

3. Download and deploy the latest version for the Sync 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 Sync 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 Sync Management Console's `web.xml` to `webapps/Sync console/WEB-INF/web.xml`.

5. Start the servlet container.

Updating the Synchronization Server

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

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

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

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

The update also upgrades the Password Synchronization Agent plug-in to its latests version automatically. Any software updates to the PSA plug-inwill be included with the new Synchronization Server zip file.

To Update the Directory Proxy Server

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

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

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

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

```
$ UnboundID-Sync-new/setup --version UnboundID® Synchronization Server 3.5.1.0
Build 2011043200609Z Revision 9235
```

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

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

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

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

```
$ view <server-root>/history/1272307020420-3.2.0.0.9235/update.log
```

Reverting an Update

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

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

To Revert to the Most Recent Server Version

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

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

Installing a Redundant Failover Server

The UnboundID® Synchronization Server supports multiple redundant failover servers that automatically become active when the main synchronization server is down for any reason. Only one Synchronization Server instance is active at any time, but multiple redundant servers can be present in the topology in a configurable prioritized order.

Before you install a redundant failover server, you must have already installed and configured a Synchronization Server instance. When installing the redundant server, the installer will copy the first Synchronization Server's configuration, including external server setup, sync pipes, sync classes, DN and attribute maps.

Note: It is critical that the Synchronization Servers (primary and secondary) have their configuration remain identical. Both servers should be registered to the "all servers" group. All `dsconfig` changes need to be applied to the server group "all servers".

To Install a Redundant Server

Before you install the redundant failover server, you should already have an existing Synchronization Server instance configured and running.

1. Unpack the UnboundID® Synchronization Server zip build. Make sure you name the unpacked directory to something other than the first server instance directory.

   ```
   $ unzip UnboundID-Sync-<version>.zip -d sync2
   ```

2. Go to the installation directory if you are not already there.

3. Follow steps 2–12 in *Installing the UnboundID Synchronization Server*, except in step 4, type `yes` to add the server to an existing topology. If you are using the `setup` tool in non-interactive mode, use the following command:

   ```
   $ ./setup --localHostName sync2.example.com --ldapPort 8389 \
     --masterHostName sync1.example.com --masterPort 7389 \
     --masterUseNoSecurity --acceptLicense --rootUserPassword password \
     --no-prompt
   ```

   The secondary server is now ready to take over as a primary server in the event of a failover. As a result, no `realtime-sync` invocations are needed for this server.

4. Verify the configuration by using the `bin/status` tool. Note the Priority Index associated with each Synchronization Server instance. The Synchronization Server with the lowest priority-index number has the highest priority.

```
$ bin/status --bindPassword secret

 ...(status output)...

          --- Sync Topology ---
Host:Port                           : Status      : Priority
------------------------------------:-------------:---------------
sync1.example.com:389 (this server)  : Active      : 1
sync2.example.com:389                : Unavailable : 2
```

5. Obtain the name of a particular Synchronization Server, run the `dsconfig` tool with the `list-external-servers` option.

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

6. To change the Priority Index of the Synchronization Server, use `bin/dsconfig`.

```
$ bin/dsconfig set-external-server-prop \
  --server-name intra-sync-sync2.example.com:389 \
  --set sync-server-priority-index:1
```

Note: To change the priority index interactively, use `bin/dsconfig`. First, enable the Advanced Objects menu. Next, on the UnboundID® Synchronization Server configuration console main menu, select External Server, select View and Edit, then select the Synchronization Server instance. Finally, on the Synchronization Server External Server menu, select the `sync-server-priority-index` property and change it to a value of your choice. Remember, the lower priority-index number has the higher priority (e.g., "1" has the highest priority).

Removing a Redundant Server

Administrators can remove a redundant server from your synchronization topology using the uninstall command on the Synchronization Server that you plan to remove from the topology. The `uninstall` command internally removes all references to the server on the other peer servers in the topology.

In the rare case that you removed a server from the topology and no longer have access to it, for example, because it got deleted from the filesystem, and the other servers in the topology still have references to it, you can run the `remove-defunct-sync-server` tool on each machine to remove the reference to the original server.

To Remove a Redundant Server

- Run the `uninstall` command on the server that you want to remove from the topology.

```
$ <server-root>/uninstall
```

Chapter 3 Configuring the Synchronization Server

The UnboundID® Synchronization Server provides a comprehensive suite of command-line tools and a graphical Sync Management Console that accesses the underlying Synchronization Server configuration framework. The configuration is stored as a flat file (LDIF format) in the `cn=config` branch. Administrators can use the server's tools to configure a single server instance or server groups remotely or locally. All configuration changes to the server and their equivalent reversion commands are recorded in the `config-audit.log`.

Before setting up the Synchronization Server, review the section *Configuration Model* to read about the important components of the Synchronization Server.

This chapter presents the following topics:

Topics:

- *Pre-Deployment Checklist*
- *Creating Administrators*
- *About the Configuration Tools*
- *About the Sync User Account*
- *Configuring the Synchronzation Server in Standard Mode*
- *Configuring the Synchronization Server Using the Management Console*
- *Configuring the Server Using dsconfig*
- *Configuring the Synchronization Server Using dsconfig*
- *Generating a Summary of Configuration Components*
- *Preparing the Synchronization Server for External Server Communication*
- *Preparing External Servers: If the Admin Does Not Have Root Access on DSEE External Servers*
- *Using Resync on the Synchronization Server*
- *Controlling Real Time Synchronization*
- *Configuring Attribute Maps*
- *Configuring the Directory Server Backend for Synchronizing Deletes*
- *Configuring DN Maps*
- *Configuring Fractional Replication*
- *Managing Failover Behavior*
- *Managing Synchronization Server Extensions*

Pre-Deployment Checklist

Prior to any deployment, you must determine the configuration parameters necessary for your Synchronization topology. Answer the following questions and record them prior to configuring your Synchronization Server instance(s).

External Servers

External Server Type. What type of external servers are you using in the Synchronization topology: UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Sun Directory Server (5.x and above), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Sun Directory Server Enterprise Edition (DSEE 6.x, 7.x), Microsoft Active Directory, Oracle (10g, 11g), Microsoft SQL Server (2005, 2008).

LDAP Connection Settings. What is the host, port, bind DN, and bind password for each external server instance(s) that you want included in the Synchronization topology?

Security and Authentication Settings. If the external server instance uses a secure connection, does it use SSL or StartTLS? What authentication method does the external server use: none, simple, external (i.e., SASL mechanisms)? If you are synchronizing to or from an Active Directory system, you must establish an SSL or StartTLS connection to the Synchronization Server.

Sync Pipes

A Sync Pipe defines a single synchronization path between the Source and Destination targets. You will need one Sync Pipe for each point-to-point synchronization path that you define for a topology of source servers to a topology of destination servers. Answer the following questions.

Sync Source. Which external server is the Sync Source for the Synchronization topology? You can define a priority order if more than one external server is defined as a Sync Source for failover purposes.

Sync Destination. Which external server is the Sync Destination for the Synchronization topology? You can define a priority order if more than one external server is defined as a Sync Destination for failover purposes.

Sync Classes

For each Sync Pipe defined, you must define one or more Sync Classes. A Sync Class defines how attributes and DNs are mapped and how Source and Destination entries are correlated. Questions required to define a Sync Class are as follows:

Evaluation Ordering. If you will be defining more than one Sync Class, what is the evaluation order of each Sync Class?

Sync Classes are evaluated according to the evaluation-order-index property and the criteria used to identify the first matching Sync Class. When there is an overlap between criteria used to identify a Sync Class, the Sync Class with the most specific criteria will be used first.

Base DNs. Are entries in the Sync Class only under specific base DNs?

Include Filters. What are the search filters to be used to search for entries in the Sync Source?

Synchronized Entry Operations. Which types of operations on entries should be synchronized: creates, modifications, and/or deletes?

DNs. What are the differences between the DNs from the Sync Source topology to the Sync Destination topology? Are there structural differences in terms of the Directory Information Tree (DIT) between the Sync Source and the Sync Destination? For example, does the Sync Source use a Nested DIT versus a Flattened DIT? Does the Sync Destination use a corresponding DIT as the Sync Source (i.e., a Nested DIT versus a Flattened DIT)?

Destination Correlation Attributes. Correlation attributes are *important* configuration parameters that are used to associate a source entry to a destination entry during the synchronization process. During the Sync configuration setup, administrators define one or more comma-separated lists of destination correlation attributes that are used to search for the corresponding source entry. The Synchronization Server first maps all attributes in a detected change from source to destination attributes using the attribute maps defined in the Sync Class. Then, it correlates the source entry to the destination entry.

The correlation attributes are flexible enough so that you can try several destination searches with different combinations of attributes until it finds the single entry that it matches. For LDAP server endpoints, you can use the distinguished name (DN) to correlate entries eventhough DN is not technically an attribute of an entry. For instance, you could specify the attribute lists "`DN,uid`", "`uid,employeeNumber`" and "`cn,employeeNumber`" to correlate entries in LDAP deployments. The Synchronization Server will search for a corresponding entry that has the same `dn` and `uid` values. If the search fails, it then searches for `uid` and `employeeNumber`. Again if the search fails, it searches for `cn` and `employeeNumber`. If none of these searches are successful, the synchronization change would be aborted and a message logged.

To prevent incorrect matches, the most restrictive attribute lists—those that will never match the wrong entry—should be first in the list, followed by less restrictive attribute lists, which will only be used when the earlier lists fail. For LDAP-to-LDAP deployments, we recommend that DN not be used as a sole correlation attribute. It is best to use DN with a combination of other unique identifiers in the entry (e.g., `dn` and `uid`) to guarantee correlation. For other non-LDAP deployments, administrators need to determine the attributes that can be synchronized across the network.

An important question related to destination correlation attributes is: Which set of Sync Destination attributes in an entry should be used to correlate an entry in the Sync Source? In other words, how does the Synchronization Server find the destination entry that corresponds to the source entry that needs to be synchronized?

Attributes. What are the differences between the attributes from the Sync Source to the Sync Destination? Some questions related to attributes are as follows:

- **Automatically Mapped Source Attributes**. Are there attributes that can be automatically synchronized with the same name at the Sync Source to Sync Destination? For example, can you set direct mappings for `cn`, `uid`, `telephoneNumber`, or for all attributes?
- **Non-Auto Mapped Source Attributes**. Are there some attributes that should not be automatically mapped from the Sync Source to Sync Destination? For example, the Sync Source may have an attribute, `employee`, while the Sync Destination may have a corresponding attribute, `employeeNumber`. If an attribute is not automatically mapped, then an Attribute Mapping must be provided if it is to be synchronized.
- **Attribute Mappings**. How are attributes mapped from the Sync Source to the Sync Destination? (For example, are they mapped directly, mapped based on attribute values, or mapped based on attributes that store DN values?)

Creating Administrators

The UnboundID Sync Management Console does not persistently store any credentials for authenticating to the Synchronization Server but uses the credentials provided by the user when logging in. When managing multiple synchronization server instances, the provided credentials must be valid for each instance. Therefore, assuming you have multiple synchronization servers —the main server and a failover—if you change an admin user on the main synchronization server instance, you must make the same change on the other server instance. Likewise, if you have multiple synchronization servers, you must make any changes manually at each server instance.

To Create an Administrator

1. To log into the console, you can either use a root user DN or create a new administrator user ID. The `dsframework` command can be used to create a user ID, for example:

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

2. Once you have set up a new admin account, the administrator can log in to the Sync Management Console using the user ID short form "someAdmin" or the full DN, "cn=someAdmin,cn=Administrators,cn=Admin Data".

About the Configuration Tools

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

- **Using the Sync Management Console**. The UnboundID® Synchronization Server provides a web-based console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All

configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.

- **Synchronization Command-Line Tools**. The UnboundID® Synchronization Server provides three command-line tools, `create-sync-pipe-config`, `resync`, and `realtime-sync` tools to quickly configure a Synchronization Server topology. The `create-sync-pipe-config` tool is a configuration wizard that guides you through a Synchronization Server configuration, records the configuration in a batch file (`<server-root>/sync-pipe-cfg.txt`), and allows you to apply the batch file to a local Synchronization Server configuration. The batch file can be re-applied to other servers. The `resync` tool is used to verify that everything is in-sync after synchronization has started or used in bulk synchronization mode to initially populate a target directory or database. The `realtime-sync` tool is used to start synchronization immediately, at a specified point at a change log event, or at a specified time duration ago.
- **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`.
- If you are configuring a Sync Pipe from scratch, we recommend using the `create-sync-pipe-config` tool as it will lead you through the steps necessary to define each Sync Pipe component.

About the Sync User Account

During the configuration process, the Synchronization Server sets up a Sync User Account DN on each external server. The account (by default, cn=Sync User) is used exclusively by the Synchronization Server to communicate with the endpoint external servers. The entry is important in that it contains the credentials (DN and password) used by the Synchronization Server to access the source and target servers. The Sync User account resides in different entries depending on the targeted system:

- For UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), the Sync User Account resides in the configuration entry (e.g., `cn=Sync User, cn=Root DNs,cn=config`).
- For Sun Directory Server and Sun DSEE, the Sync User account resides under the base DN in the userRoot backend (e.g., cn=Sync User,dc=example,dc=com). We also recommend that the Sync User account NOT be in the cn=config branch for Sun Directory Server and DSEE machines. If it resides there, delete it, and then add it to the normal backend (dc=example,dc=com) and update the configuration in the Synchronization Server.
- For Microsoft Active Directory servers, the Sync User account resides in the Users container (e.g., `cn=Sync User,cn=Users,DC=adsync,DC=unboundid,DC=com`).
- For Oracle and Microsoft SQL Servers, the Sync User account is a login account (SyncUser) with the sufficient privileges (for example, Resource and Connect) to access the tables to be synchronized.

Although in most cases, modifications to this account will never take place, you can ensure that the entry never gets synchronized by setting up an optional Sync Class if your Sync User account resides in the userRoot backend (Sun Directory Server or Sun DSEE) or Users container (Microsoft Active Directory). For example, you can configure this Sync Class to have all CREATE, MODIFY, and DELETE operations set to false, so that the Sync User Account never gets synchronized with the other user entries.

Configuring the Synchronzation Server in Standard Mode

The general process to configure a Synchronization Server (standard mode) is to first define the external servers in the topology and then define the Sync Pipe(s) and its associated Sync Classes. You can use the `create-sync-pipe-config` tool to set up your Sync Pipes and Sync Classes. For bidirectional deployments, you will need to configure two Sync Pipes, one for each directional path (Sun DS 5.2 to UnboundID DS and vice-versa).

In the following example, we will also set up a simple attribute map that maps an email attribute on the first endpoint servers to the mail attribute on the second endpoint servers. In typical cases like these, you need to set up a specific attribute mapping from email to mail for the Sun to UnboundID Sync Pipe and also have both of these source attributes be excluded for automatic mapping. If you do not exclude the source attribute, the Synchronization Server will attempt to create an email attribute on the second endpoint topology, which could fail if the email attribute is not present in the destination server's schema. Conversely, you have to create a specific mail to email mapping and auto-exclude the source attribute on the UnboundID to Sun Sync Pipe going the other direction.

For this example, you will define two sync classes: one to handle the customized email to mail attribute mapping; the other, to handle all other cases (called the default sync class). Next, you will use the `dsconfig` command to create the specific attribute mapping. After that, you can run the `resync` command to test the mappings. Finally, you can start synchronization using the `realtime-sync` command.

Assumptions

The following example shows a bidirectional synchronization deployment in standard mode between a Sun Directory Server 5.x topology and an UnboundID Directory Server topology. The example assumes that you have two replicated topologies configured: the first endpoint topology consists of two Sun Directory Server LDAP servers (version 5.2 patch 4): the main server and one failover. Both Sun Directory Servers 5.x have their Retro Change logs enabled and contains the full DIT that will be synchronized to the second endpoint. The second endpoint topology consists of two UnboundID Directory Servers (version 3.x): the main server and one failover. Both UnboundID Directory Servers have their change logs enabled and contain entries similar to the first endpoint servers, except that it uses a mail attribute, instead of an email attribute.

Note: For UnboundID Directory Server and Alcatel-Lucent 8661 Directory Server systems, you must configure the `changelog-deleted-entry-include-attribute` property on the change log backend. This property

allows for the proper synchronization of DELETE operations that occur with this endpoint server. For more information, see *Configuring the Directory Server Backend for Synchronizing Deletes*.

Configuring the Synchronization using create-sync-pipe-config

For all configurations, we strongly recommend that you use the `create-sync-pipe-config` command-line wizard to guide you through a Sync Pipe configuration. Once the configuration is completed, you can fine-tune the settings using the `dsconfig` tool.

To Configure the Synchronization Server using create-sync-pipe-config

1. Start the Synchronization Server.

```
$ <server-root>/bin/start-sync-server
```

2. From the bin directory, use the `create-sync-pipe-config` tool to set up the Synchronization sync pipes. The tool will start the command-line wizard and walk you through the steps to configure your Sync Pipes.

```
$ bin/create-sync-pipe-config
```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** (yes) to continue the configuration.

4. On the Synchronization Mode menu, press **Enter** to select Standard mode. A standard Mode Sync Pipe will fetch the full entries from both the source and destination and compare them to produce the minimal set of changes to bring the destination into sync. A notification mode Sync Pipe will skip the fetch and compare phases of processing and simply notify the destination that a change has happened and provide it with the details of the change. Notifications are currently only supported from UnboundID and Alcatel-Lucent 8661 Directory or Proxy Servers 3.0.3 or later.

5. On the Synchronization Directory menu, select if the Synchronization topology will be one-way (1) or bidirectional (2). In this example, enter the number for bidirectional. If you typed the option for one-way synchronization, you will next see the Source Endpoint Type menu. For this example, because you entered the option for bidirectional synchronization, you will next see the First Endpoint Type menu.

6. On the First Endpoint Type menu, select the directory or database server for the first endpoint. The available options are seen below. In this example, type the number corresponding to the Directory Server.

```
>>>> First Endpoint Type
Enter the type of data store for the source endpoint:
    1)  UnboundID Directory Server
    2)  UnboundID Proxy Server
    3)  Alcatel-Lucent Directory Server
    4)  Alcatel-Lucent Proxy Server
    5)  Sun Directory Server
    6)  Microsoft Active Directory
    7)  Microsoft SQL Server
    8)  Oracle Database
```

```
    9)  Custom JDBC

    b)  back
    q)  quit

Enter choice [1]: 5
```

7. On the First Endpoint Name menu, type a name for the Endpoint Server, or accept the default ("Sun Directory Server"). In this example, type "Sun DS 5.2".

8. On the Base DNs menu, type the base DN on the first endpoint topology where the entries will be searched. In this example, accept the default (`dc=example,dc=com`). If you have other base DNs, type the DN or press **Enter** when finished. If you enter another base DN, make sure that it does not overlap with the other base DN(s).

9. On the Server Security menu, select the server security type. The available options are None (LDAP), SSL, and StartTLS. In this example, press **Enter** to accept the default (None).

10. On the First Endpoint Servers menu, type the host name and listener port number for the First Endpoint Server, or accept the default (port 389). Make sure that the endpoint servers are online and running. The server will perform a test connection to the server. If the server is unresponsive, you will be asked to retry contacting the server, discard the server, or keep the server.

11. After entering the first server, enter the hostname and listener ports of the additional servers in the endpoint topology. The server will also perform a test connection to this server. If the server is unresponsive, you will be asked to retry contacting the server, discard the server, or keep the server. At this stage, you can enter more servers, remove the existing servers, or press **Enter** when you are finished entering the servers.

12. Next, you will be prompted to enter the Sync User account DN for the endpoint servers. This step will ask you to enter a Sync User Account DN (`cn=Sync User,cn=Root DNs,cn=config`) and password.

13. At this point, you must set up the servers in the Second Endpoint topology. Repeat steps 6–12 to configure the second endpoint server. Select the option for UnboundID, and then set up the two external endpoint servers and Sync User Account DN.

Prepare the External Servers

1. After you have configured the first and second endpoint topologies, the Synchronization Server will prompt you to "prepare" each external server by testing the connection to each server. This step entails determining if each external server has the necessary privileges (e.g., root privileges are required) to communicate and to transfer data during synchronization. If an error occurs, the Synchronization Server will prompt you to re-configure the specific connection parameter. Using the Sync User Account DN, the server verifies the base DNs, and enables and checks the change log on the external server. If the maximum age of the change log has not been set, you will also be prompted for a value between two hours or seven days (the recommended maximum age is 2 days).

2. Repeat step 1 to prepare the other external servers.

Note: If your endpoint servers have no base entries or data, the command cannot create the `cn=Sync User,cn=Root DNs,cn=config` account. In this specific case, you can select 2 (Abandon the Operation) to continue, then create the base entry on the destination servers.

Configure the Sync Pipes and its Sync Classes

1. Continuing the `create-sync-pipe-config` session, you will be prompted to create a name for the Sync Pipe on the Sync Pipe Name menu. Type a descriptive name to identify the Sync Pipe or accept the default. Because this example is bidirectional, the following step is setting up a Sync Pipe path from the Sun DS 5.2 endpoint to the UnboundID Directory Server endpoint. In a later step, you will need to define another Sync Pipe from UnboundID DS to Sun DS.

2. On the Sync Class Definitions menu, type `Yes` if you want to create a custom Sync Class or press **Enter** to accept the default (no). A Sync Class defines the operation types (e.g., creates, modifies, or deletes) and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. In this example, create a basic sync class for the `email` to `mail` attribute mapping, which will exclude the source attribute from automatic synchronization. Later in the procedure, you will need to configure the `email` to `mail` attribute mapping using the `dsconfig` tool.

3. Next, you will be prompted to create a Sync Class name. Enter a name for the new Sync Class. For this example, enter "SunDS>UBID".

4. On the Base DNs for Sync Class menu, enter one or more base DNs if you want to synchronize specific subtrees of a DIT. Entries outside of the specified base DNs will be excluded from synchronization. Make sure the base DNs do not overlap in any way. In this example, press **Enter** to accept the default (no) as we will not restrict any entries during the synchronization process.

5. On the Filters for Sync Class menu, you can define one or more LDAP search filters to restrict specific entries for synchronization. Those entries that do not match the filters will be excluded from synchronization. In this example, press **Enter** to accept the default (no).

6. Next, on the Synchronized Attributes for Sync Class menu, specify which attributes will be automatically mapped from one system to another. You can select the following options: 1 to Synchronize all attributes, 2 to Specify attributes to synchronize, 3 to Specify attributes to exclude from synchronization. In this example, assume that the Sun Directory Server endpoint has an `email` attribute that needs to be mapped to a `mail` attribute in the target endpoint servers. A specific attribute mapping will be configured in a later step. In this example, we will exclude the source attribute (email) from being auto-mapped to the target servers by selecting the option, `Specify attributes to exclude from synchronization`.

7. On the Operations for Sync Class menu, select the operations that will be synchronized for the Sync Class (1 for creates, 2 for deletes, 3 for modifies, 4 for none), or press **Enter** to accept the default ("1, 2, 3"). You can enter a comma-separated list of numbers that

correspond to the operation. For these example, press **Enter** to accept the default (creates, deletes, modifies).

8. Next, define a default or "catch-all" Sync Class that specifies how the other entries are processed. In the following example, press **Enter** to continue, the system will create a Sync Class called "Default Sync Class".

9. On the Default Sync Class Operations menu, specify the operations that the default Sync Class (1 for creates, 2 for deletes, 3 for modifies, 4 for none) will handle during synchronization. In this example, press **Enter** to accept the default (1, 2, 3). You have successfully defined one sync pipe that goes from Sun Directory Server to UnboundID Directory Server.

10. At this stage, you must define a Sync Pipe going from the UnboundID Directory Server to the Sun Directory Server. Repeat the previous steps 4–9. When you create a sync class, make sure to create a UBID>SunDS sync class, and then exclude the `mail` attribute from being synchronized to the other endpoint servers.

Review the Configuration and Apply the Changes

1. Review the Sync Pipe Configuration Summary, and then, press **Enter** to accept the default ("write configuration"), which records the commands in a batch file (`sync-pipe-cfg.txt`). The batch file can be re-used to set up other Sync topologies.

```
  >>>> Configuration Summary
              Sync Pipe: Sun DS 5.2 to UnboundID DS
               Source: Sun DS 5.2
                  Type: Sun Directory Server
                 Access Account: cn=Sync User,cn=Root DNs,cn=config
                 Base DN: dc=example,dc=com
                 Servers: sun-ds1.example.com:21389, sun-ds2.example.com:22389
                Destination: UnboundID DS
                 Type:  UnboundID Directory Server
                 Access Account: cn=Sync User,cn=Root DNs,cn=config
                 Base DN: dc=example,dc=com
                 Servers: UnboundID.example.com:23389, UnboundID.example.com:24389
                Sync Classes:
                 SunDS>UBID
                   Base DN:
                   Filters:
                    DN Map: None
                   Synchronized Attributes: all except: email
                   Operations: Creates,Deletes,Modifies
                  DEFAULT
                   Operations: Creates,Deletes,Modifies
              Sync Pipe:  UnboundID DS to Sun DS 5.2
               Source:  UnboundID DS
                  Type:  UnboundID Directory Server
                 Access Account: cn=Sync User,cn=Root DNs,cn=config
                 Base DN: dc=example,dc=com
                 Servers:  UnboundID.example.com:23389, UnboundID.example.com:24389
                Destination: Sun DS 5.2
                 Type: Sun Directory Server
                 Access Account: cn=Sync User,cn=Root DNs,cn=config
                 Base DN: dc=example,dc=com
                 Servers: sun-ds1.example.com:21389, sun-ds2.example.com:22389
                Sync Classes:
                 UBID>SunDS
                   Base DN:
                   Filters:
                    DN Map: None
                   Synchronized Attributes: all except: mail
                   Operations: Creates,Deletes,Modifies
                  DEFAULT
                   Operations: Creates,Deletes,Modifies
                w)  write configuration
```

```
              b)  back
              q)  quit
            Enter choice [w]:
```

2. Apply the configuration changes to the local Synchronization server instance using a dsconfig batch file. Once you have applied the changes to the server, you can review the configuration in the `<server-root>/sync-pipe-cfg.txt` file.

3. If you have any Server SDK extensions, save them to the `<server-root>/lib/extensions` directory.

4. Connect to the Synchronization Server using the LDAP Connection Parameters: host name, port, user bind DN and bind DN password. The configuration is recorded to the `<server-root>/sync-pipe-cfg.txt`.

   You have successfully configured the initial Sync Pipes for your system. The next step will be to configure the attribute mappings using the `dsconfig` command.

Configure the Attribute Map and Mapping

The following section continues from the previous example by defining an attribute map that has a mapping from the email attribute in the source servers to a mail attribute in the target servers. You must ensure that both attributes are valid in the target servers and are present in their respective schemas.

1. On the Synchronization Server, run the `dsconfig` command to create an attribute map for the "SunDS>UBID" sync class for the "Sun DS 5.2 to UnboundID DS" sync pipe, and then run the second `dsconfig` command to apply the new attribute map to the Sync Pipe and Sync Class.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "SunDS>UBID Attr Map" \
  --set "description:Attribute Map for SunDS>UBID Sync Class" \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret

$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Sun DS 5.2 to UnboundID DS" \
  --class-name "SunDS>UBID" \
  --set "attribute-map:SunDS>UBID Attr Map" \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

Note: You can use `dsconfig` in interactive mode. The attribute map and attribute mapping options appear in the Synchronization Server Configuration Console main menu.

2. Next, create an attribute mapping (from `email` to `mail`) for the new attribute map.

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "SunDS>UBID Attr Map" --mapping-name mail --type direct \
  --set "description:Email>Mail Mapping" --set from-attribute:email \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

3. Because this example shows how to set up a bidirectional deployment, repeat steps 1–2 to create an attribute map for the UBID>SunDS sync class for the UnboundID DS to Sun DS 5.2 sync pipe, and create an attribute mapping that maps `mail` to `email`.

```
$ bin/dsconfig --no-prompt create-attribute-map --map-name "UBID>SunDS Attr Map" \
  --set "description:Attribute Map for UBID>SunDS Sync Class" \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret

$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "UnboundID DS to Sun DS 5.2" --class-name "UBID>SunDS" \
  --set "attribute-map:UBID>SunDS Attr Map" \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret

$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "UBID>SunDS Attr Map" --mapping-name email --type direct \
  --set "description:Mail>Email Mapping" --set from-attribute:mail \
  --port 7389 --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure Server Locations

The Synchronization Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the Sync Server prefers to connect to endpoint servers in the same location as itself and also prefers to failover to endpoint servers in the same location as itself. If there are no location settings configured, then the Synchronization Server will simply iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

It is good practice to set the `location` property on the external servers and the `location` property of the Synchronization Server global configuration whenever possible. For more information on failover location preference, see the section *Failover Server Preference*.

Note: Location-based failover is only applicable for LDAP endpoint servers, such as the UnboundID Directory Server of the UnboundID Directory Proxy Server.

1. On the Synchronization Server, run `dsconfig` to set the location for each external server in the Sync Source and Sync Destination. For example, the following command sets the location for six servers in two data centers, "austin" and "dallas".

```
$ bin/dsconfig set-external-server-prop --server-name example.com:1389 \
  --set location:austin
$ bin/dsconfig set-external-server-prop --server-name example.com:2389 \
  --set location:austin
$ bin/dsconfig set-external-server-prop --server-name example.com:3389 \
  --set location:austin
$ bin/dsconfig set-external-server-prop --server-name example.com:4389 \
  --set location:dallas
$ bin/dsconfig set-external-server-prop --server-name example.com:5389 \
  --set location:dallas
$ bin/dsconfig set-external-server-prop --server-name example.com:6389 \
  --set location:dallas
```

2. On the Synchronization Server, run `dsconfig` to set the location on the Global Configuration. This is the location of the Synchronization Server itself, and preferably, it will be the same as at least one of your external servers. In this example, set the location to "austin".

```
$ bin/dsconfig set-global-configuration-prop --set location:austin
```

Complete the Bidirectional Deployment

At this stage, you have configured the Sync Pipes, Sync Classes, and Attribute Mappings necessary for your synchronization topology.

1. Run the bulk synchronization command `resync` to test the attribute mapping. For more information, see *Using Resync on the Synchronization Server*. Any logging performed during a `resync` operation appears in the `logs/tools/resync.log`.

```
$ bin/resync --pipe-name "Sun DS 5.2 to UnboundID DS" \
  --sourceSearchFilter "(uid=user.0)" --dry-run \
  --logFilePath logs/tools/resync.log --logLevel debug
```

2. Finally, start the synchronization process using the `realtime-sync` command. For more information, see *Controlling Real Time Synchronization*.

```
$ bin/realtime-sync start --pipe-name "Sun DS 5.2 to UnboundID DS" \
  --pipe-name "UnboundID DS to Sun DS 5.2" --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
```

   You have successfully completed the bidirectional Synchronization deployment.

Configuring the Synchronization Server Using the Management Console

The UnboundID Synchronization Server provides a graphical web application tool, UnboundID Sync Management Console, which accesses the server's underlying configuration. The Sync Management Console provides functionally equivalent to the `dsconfig` command-line tool in addition to monitoring and server information.

Note: Like the `dsconfig` tool, all changes made using the Sync Management Console are recorded in `logs/config-audit.log`.

Configuring the External Servers Using the Management Console

External servers are the specific servers that should be included in the Synchronization topology. You must specify the LDAP connection and security parameters necessary to send requests to these servers. External servers can be either UnboundID Directory Servers, UnboundID Directory Proxy Servers (3.x), Alcatel-Lucent 8661 Directory Servers, Alcatel-Lucent 8661 Directory Proxy Servers (3.x), Sun Directory Server 5.x, Sun Directory Server Enterprise Edition (DSEE 6.x, 7.x), Microsoft Active Directory, Oracle (10g,11g), or Microsoft SQL Server (2005, 2008).

To Configure the External Servers

1. Start the Synchronization Server.

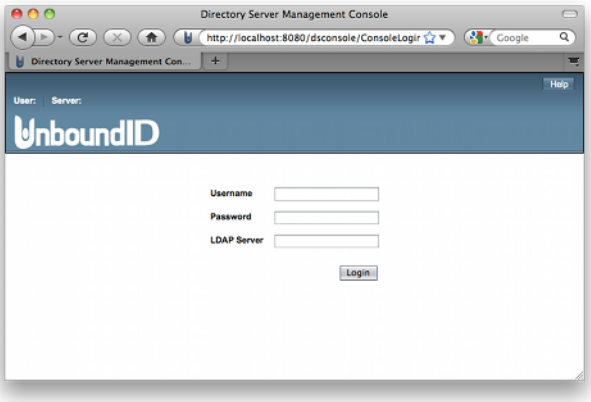
```
$ <server-root>/bin/start-sync-server
```

2. Start the servlet container.

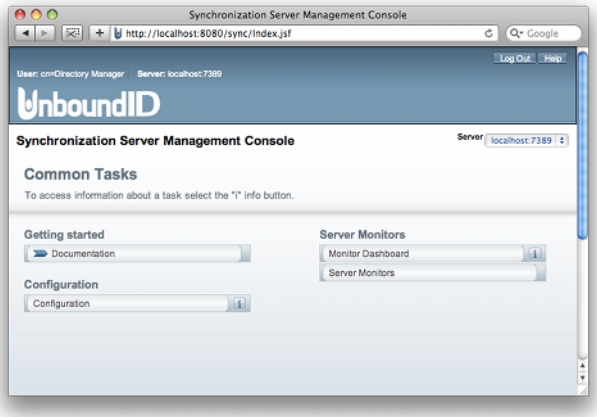
```
$ /apache-tomcat-<version>/bin/startup.sh
```

3. Open a browser to http://hostname:8080/sync. The servlet container listens on port 8080 for HTTP requests.

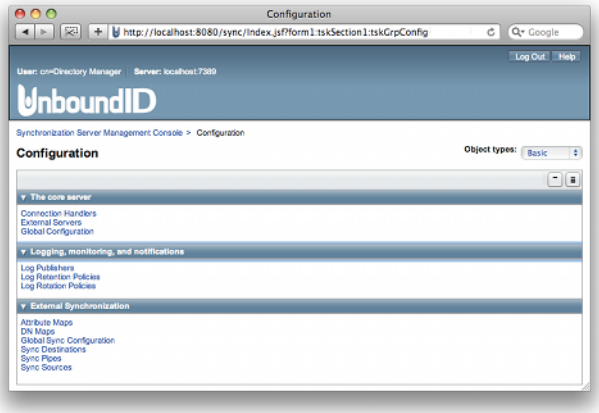
4. Type the root user DN (or any authorized administrator user name) and password, and the server hostname or IP address and port to log on (for example, server1.example.com:389).



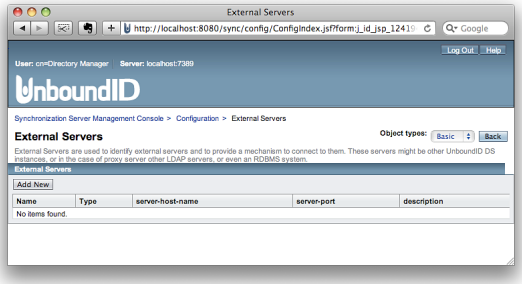
5. On the Synchronization Server Management Console, click **Configuration**.



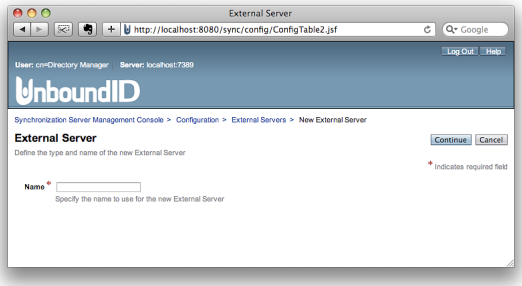
6. Under "The core server," click **External Servers** to identify all of the servers that will by synchronized.



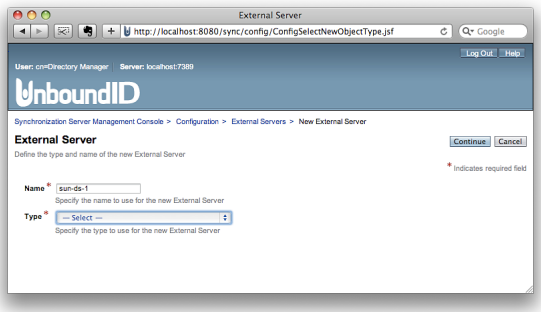
7. Click the **Add New** button to define the first server in the topology.



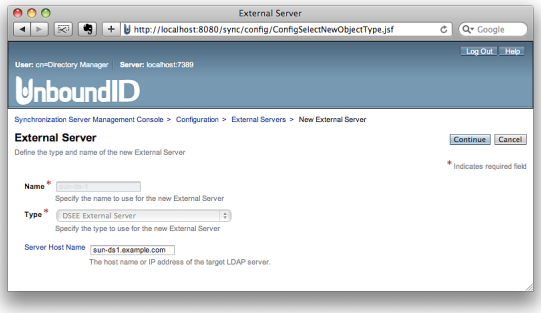
8. Type a name for the external server, and then click **Continue**. The name can be any label that will help you identify the server.



9. On the Type drop-down menu, select the type of external server that you are defining. In this example, select Sun DS Sync Source.



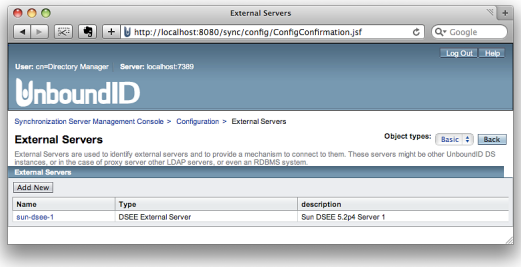
10. Type the hostname for the external server, and then click **Continue**.



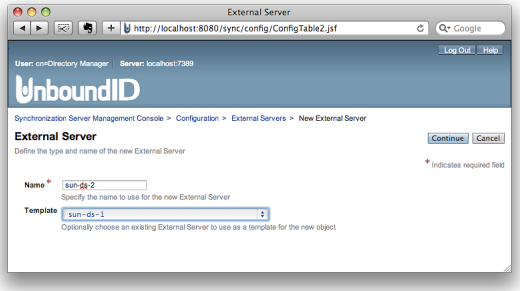
11. Type the external server's connection parameters that were configured when the server was first installed. If security and authentication settings were configured for the external server, define them on this page. When completed, click **Confirm then Save**.

12. Click **Apply** to save the settings for this external server. The equivalent `dsconfig` command-line instruction is displayed to recreate the external server in a scripted installation or to quickly define similar external servers from the command line.

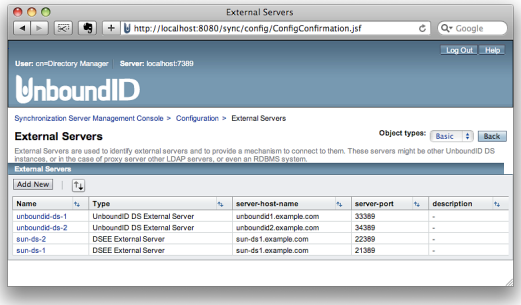
13. Click **Add New** to define another external server.



14. Once you have defined one external server, you can use the settings defined for the first server as a template for the next one. On the External Server page, type the name of the new external server, select the first server on the Template drop-down menu, and then click **Continue**. In this example, only the server name and host name has changed for the second server.



15. At this stage, repeat steps 8–13 to define the other external source and target servers. You will only need to change the description, host name, and port number for each server.



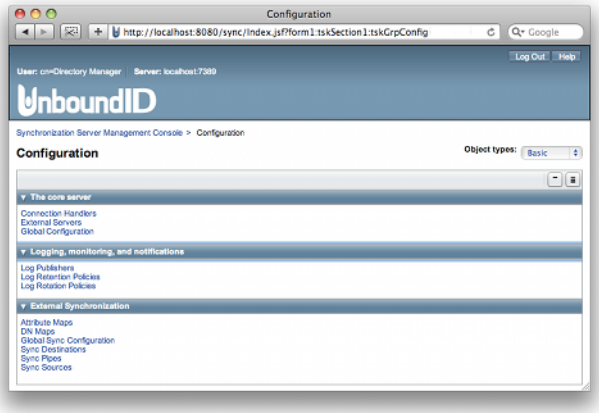
You have successfully defined the external servers in the Synchronization topology.

Configuring the Sync Pipe Using the Management Console

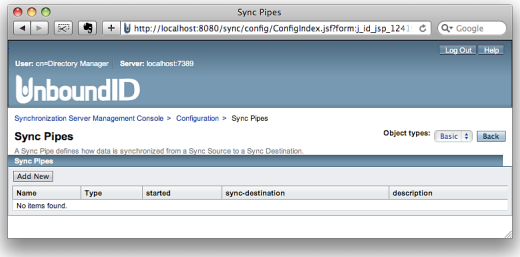
Next, you will need to configure how synchronization is processed between the Source and Destination topologies. The next two sections present information on how to set up the Sync Pipe and Sync Class.

To Configure the Sync Pipe Using the Management Console

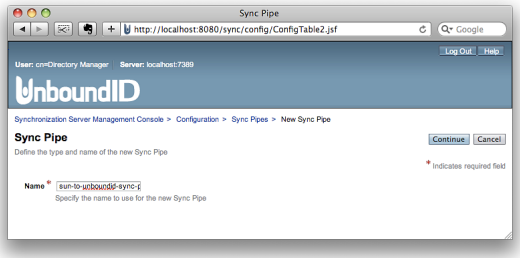
1. On the Configuration page, click **Sync Pipes**.



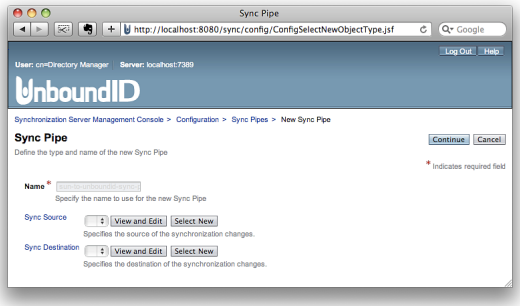
2. Click **Add New** to define a Sync Pipe.



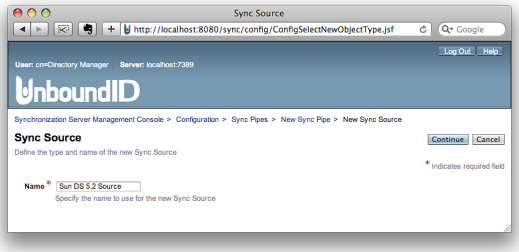
3. Type the name of the Sync Pipe, and then click **Continue**.



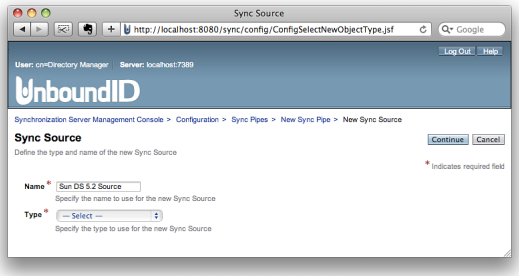
4. For the Sync Source, click **Select New**.



5. Type a name for the Sync Source to identify the topology, and then click **Continue**.

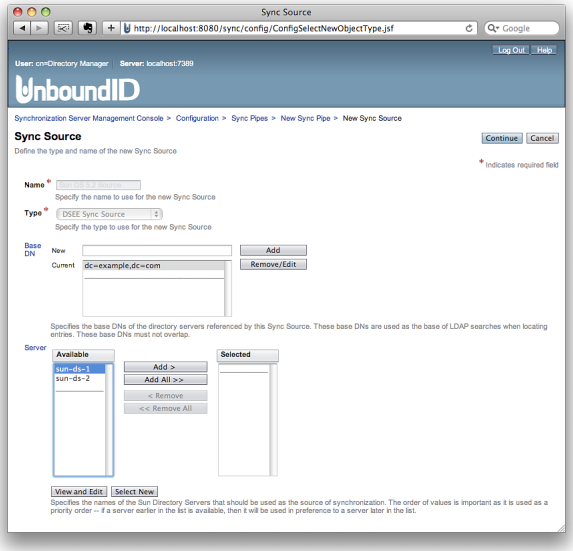


6. On the Type drop-down menu, select the Sync Source type. In this example, select "Sun DS Sync Source."

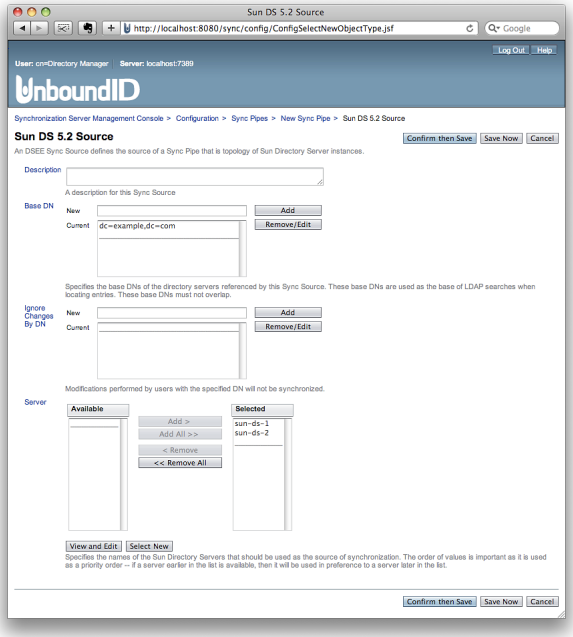


7. In the New field, type the base DN to be used for synchronization searches, and then click **Add**. The base DN defines the scope of the searches that the Synchronization Server processes for its change flow. You can specify more than one base DN, but the base DNs must not overlap another base DN (i.e., they cannot be sub-branches of another base DN).

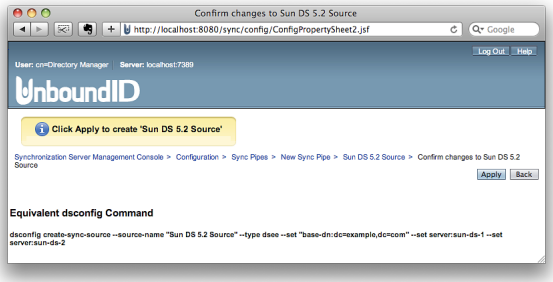
8. In the Server section, select the server(s) to be used as the Sync Source. The order of the servers is important as it determines the priority order of the Synchronization source. Specifically, the Synchronization Server will connect to the first server when detecting changes as long as it is available. Click **Add All** if the default order is acceptable. For example, in the graphic below, the sun-ds-1 server is used in preference to the sun-ds-2 server. If you want to select the second server as the higher priority, click the server link, and then click **Add**. Then, move the other server to the Select column. Click **Continue** when done.



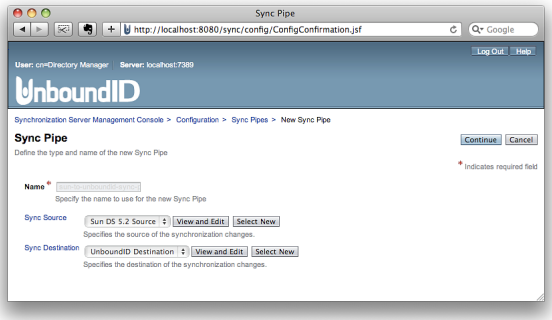
9. In the Ignore Changes by DN field, type a DN for which the Synchronization Server should ignore any modifications by the user DN, and then click Add. This function serves as a form of loopback detection from the Destination target to the Source target when using bidirectional synchronization. During loopback, the Synchronization Server ignores any modifications made by the user, except for any deletion operations. Click **Confirm then Save** when done. For example, you can specify the DN of the synchronization user, `cn=Sync User,cn=Root DNs,cn=config`.



10. Click `Apply` to save the Sync Source configuration.

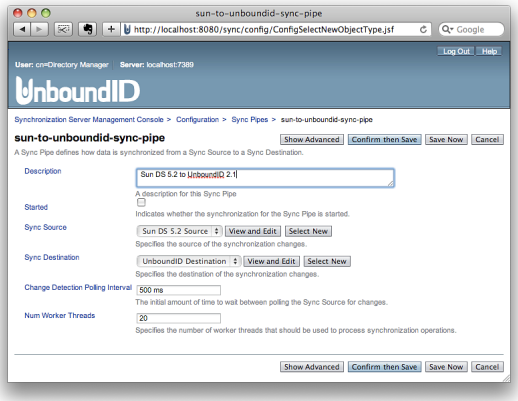


11. Repeat steps 5–10 for the Sync Destination, so that you have the Sync Source and Sync Destination defined for the Sync Pipe.

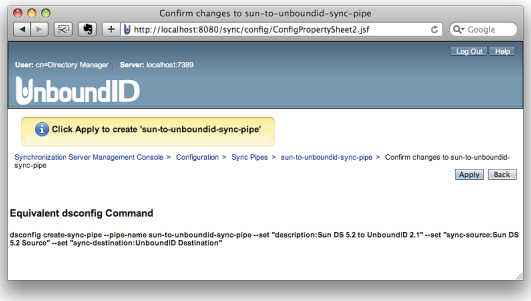


12. After defining the Sync Destination, enter a description for the Sync Pipe. Modify the Polling Interval if necessary. The Polling Interval is the amount of time that the Synchronization Server waits between checking the Sync Source for changes. The default time is 500 ms.

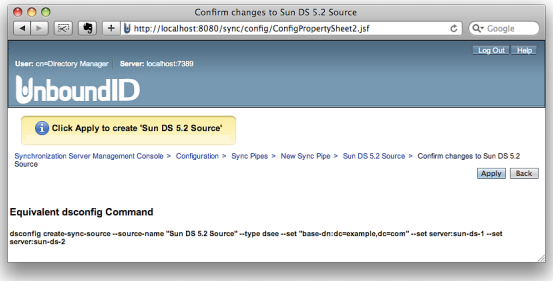
Although likely unnecessary, you can change the default number of worker threads if necessary. The number of worker threads should be increased if there is a large network latency between the Sync Source servers and the Sync Destination servers. Click **Confirm then Save** when done.



13. Click `Apply` to save the changes.



14. Repeat steps 1–13 if you want to define a bidirectional Sync pipe from the Sync Destination to Sync Source. Otherwise, click **Back** to define the next Synchronization configuration.



Configuring the Sync Class Using the Management Console

A Sync Class is defined for each type—or class—of entry that should be treated differently by the Synchronization server. This includes what types of changes are synchronized, what

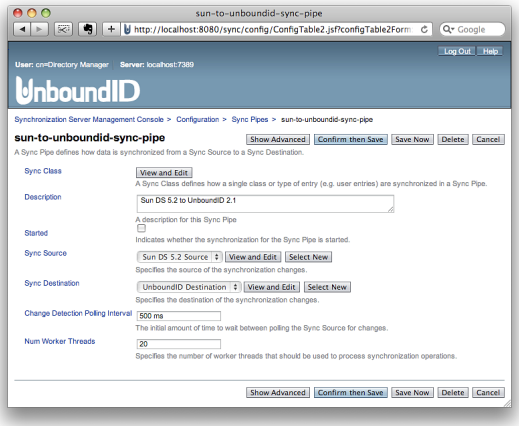
attri- butes are synchronized and how they are mapped, how source and destination entries are correlated, and how DNs are mapped.

The Sync Class also defines what attributes within the entries should be included or excluded in the synchronization process. When a change to an entry is first detected in a Sync Source, the Sync Pipe evaluates the inclusion criteria (i.e., `include-base-dn` and `include-filter`) to find the first matching Sync Class according to the `evaluation-order-index` property. If a change does not match any Sync Class, then it is discarded. Otherwise, the matching Sync Class processes any attribute or DN mappings and determines what type of change is synchronized.

Note: If you do not want certain types of entries to be synchronized, then you can define a Sync Class for these attributes and then clear the `synchronize-creates`, `synchronize-modifies`, and `synchronize-deletes` boxes.

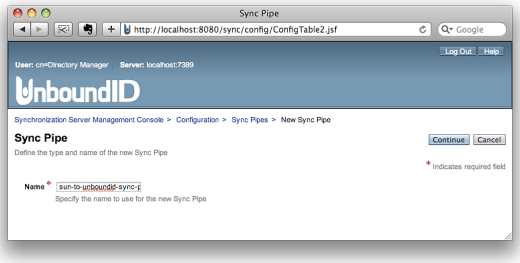
To Configure a Sync Class Using the Management Console

1. On the Sync Pipe page, click **View and Edit** next to the Sync Class that you want to configure.



2. Click **Add New** to define a Sync Class for the Sync Pipe. A Sync Pipe may have more than one Sync Class defined.

3. Type a name for the Sync Class.



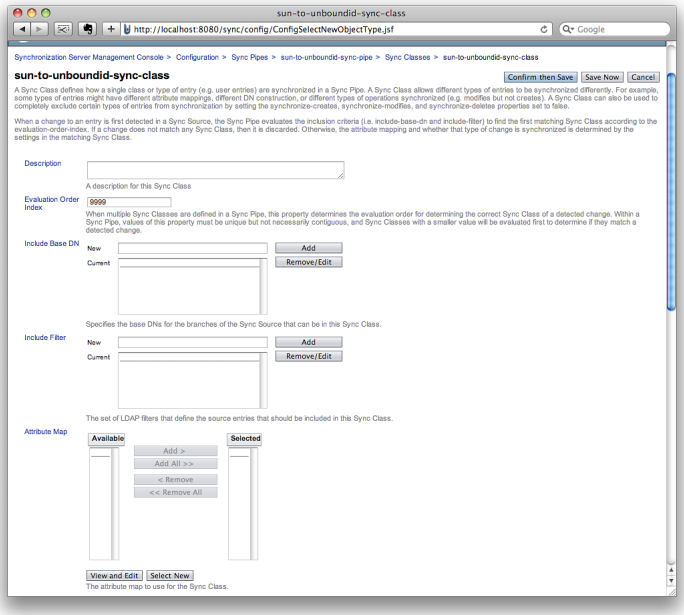
4. On the Sync Class page, in the Description field, type a general description for the Sync Class. This is an optional step.

5. In the Evaluation Order Index field, type the priority ordering for the Sync Class if you have more than one Sync Class configured for the topology. Sync Classes with a smaller evaluation order index are evaluated first. Because this example defines only one Sync Class, the default value of 9999 is used.

6. In the Include Base DN field, type the base DN for the branches of the Sync Source that contain entries in this Sync Class. Only entries with this base DN will be included in the Sync Class. This is an optional step. If no base DN is specified, the location of the entry in the Sync Source is not taken into account when determining if an entry is part of this Sync Class.

7. In the Include Filter field, type a search filter that determines which entries are in the Sync Class. If no filter is specified, all entries within the specified included base DNs are included in the Sync Class.

8. In the Attribute Map section, click **Select New** to define a set of attribute mappings from Sync Source to Sync Destination. In this example, the Sync Source (Sun DS 5.2) attributes map directly to the Sync Destination (UnboundID Directory Server), so no attribute maps require definition. See *Configuring Attribute Maps*.



9. In the DN Map section, click Select New to define a set of DN mappings from Sync Source to Sync Destination. In this example, the Sync Source DNs (Sun DS 5.2) map directly to the Sync Destination DNs (UnboundID Directory Server), so no DN maps require definition. See *Configuring DN Maps*.

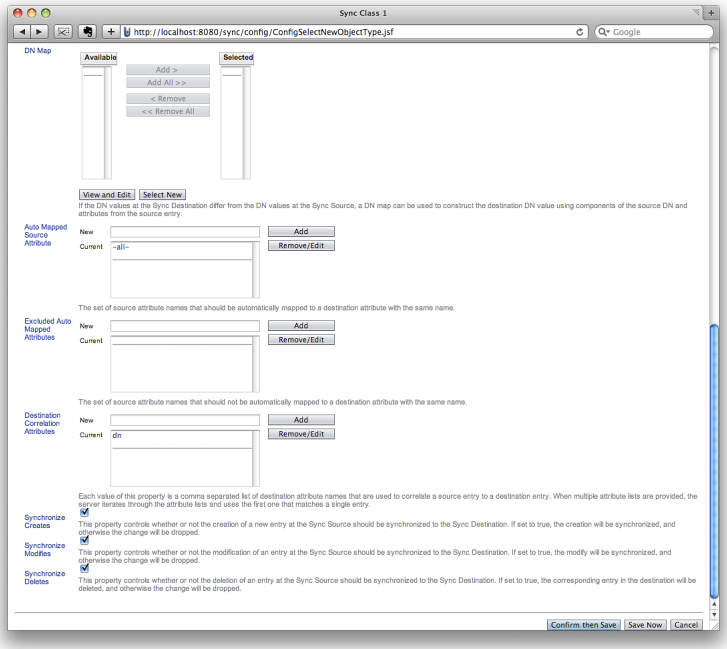
10. In the Auto Mapped Source Attribute field, type any source attributes that should be automatically mapped to attributes of the same name in the destination target, and then click **Add**. By default, all attributes are mapped automatically.

11. In the Excluded Auto Mapped Attributes field, type any source attributes that should not be automatically mapped to attributes in the destination target, and then click **Add**. By default, no attributes are excluded.

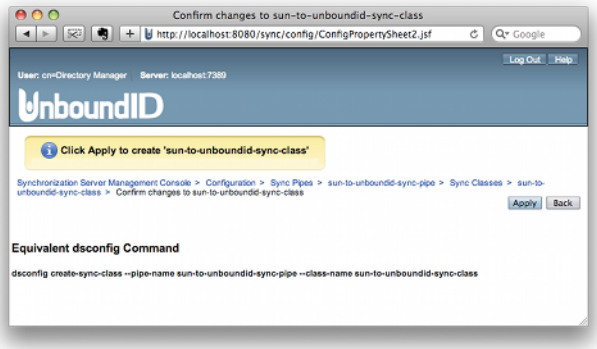
12. In the Destination Correlation Attributes field, type a comma-separated list of destination attributes that are used to correlate a source entry to a destination entry. For example, the default option is to use the DN to correlate entries (for LDAP-to-LDAP deployments), but you could specify that the `DN` and `uid` attributes be used to correlate entries, or the `cn` and `employeeNumber` attributes, or others, depending on how the entries are structured in the Sync Source and Sync Destination, respectively. To prevent incorrect matches, the most restrictive attribute lists, those that will never match the wrong entry, should be first in the list, followed by less restrictive attribute lists, which will only be used when the earlier lists fail.

13. Clear the specific types of changes that you do not want to synchronize: Synchronize Creates, Synchronize Modifies, Synchronize Deletes

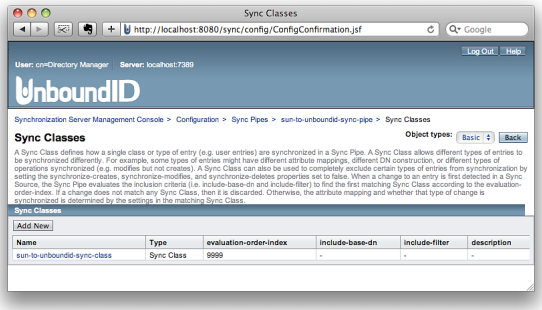
14. When completed, click **Confirm then Save**.



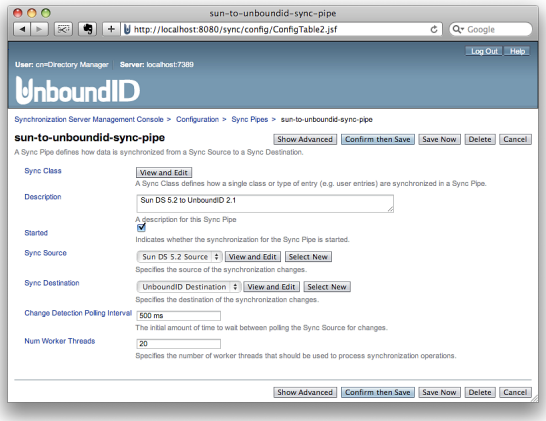
15. Click **Apply** to complete defining the Sync Class.



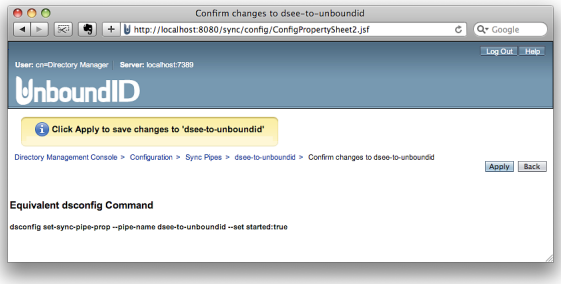
16. On the Sync Classes page, click **Back** to return to the Sync Pipe page.



17. On the Sync Pipe page, click **Started**, and then click **Confirm then Save**. The Started field on the Sync Pipe controls whether a given Sync Pipe is synchronizing.



18. Click **Apply** to save the settings.



19. Repeat steps 2–18 to create another Sync Class, or log out of the console.

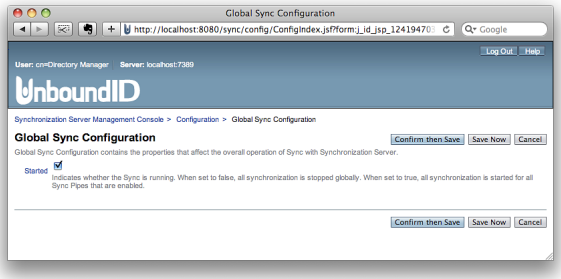
Starting the Global Sync Configuration Using the Management Console

After you have configured the Sync Pipe and Sync Class, you must start the Global Sync configuration property (that is, enable synchronization). By starting the Synchronization Server, it starts or stops synchronization for all configured Sync Pipes. Each Sync Pipe must also be started for synchronization to take place.

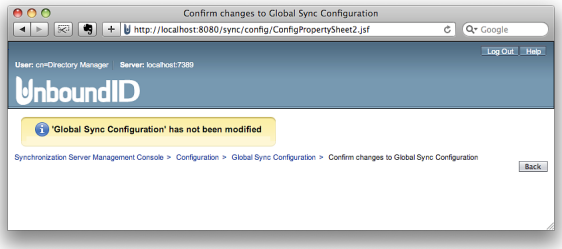
To Start the Global Sync Configuration Using the Management Console

1. On the Configuration page, click **Global Sync Configuration**.

2. Click the Started box, and then click **Confirm then Save**.



3. Click **Apply** to save the configuration settings.



4. After you have completed the configuration, run the `prepare-endpoint-server` tool from the command line to ensure that the external servers can communicate with each other. See *Preparing the Synchronization Server for External Server Communication* for more information.

5. Next, run the `bin/resync` tool to verify the synchronization configuration. See *Verifying the Synchronization Configuration using Resync*.

6. Next, run the `bin/realtime-sync` tool to start the startpoint. See *Setting Startpoints*.

Configuring the Server Using dsconfig

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

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

Using dsconfig in Interactive Command-Line Mode

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

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

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

Using dsconfig Interactive Mode: Viewing Object Menus

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

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.
- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.

- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

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

```
>>>> UnboundID® Synchronization Server configuration console main menu

What do you want to configure?

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

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

Enter choice:
```

Using dsconfig in Non-Interactive Mode

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

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

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

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

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

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

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

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

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

To Get the Equivalent dsconfig Non-Interactive Mode Command

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

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

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

Using dsconfig Batch Mode

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

```
$ bin/dsconfig --no-prompt --hostname host1 --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret  \
  --batch-file /path/to/sync-pipe-config.txt
```

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

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

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

Configuring the Synchronization Server Using dsconfig

You can use the dsconfig tool to configure any part of the Synchronization Server. However, you will likely use the tool for more fine-grained adjustments. If you are configuring a Sync Pipe for the first time, you should use the `bin/create-sync-pipe-config` tool as it will guide you through the necessary Sync Pipes creation steps for your system.

To Configure the Synchronization Server Using dsconfig Interactive

1. Launch the `dsconfig` tool in interactive command-line mode.

   ```
   $ <server-root>/bin/dsconfig
   ```

2. On the LDAP Connection Parameters menu, type the Synchronization Server host name, or IP address, or press **Enter** to accept the default.

3. On the Synchronization Server Connection menu, type the number corresponding to the type of LDAP connection type (1 for LDAP, 2 for SSL, 3 for StartTLS) that you are using on the Synchronization Server, or press **Enter** to accept the default.

4. Next, type the LDAP listener port number, and then type the user bind DN, and the bind DN password.

5. On the Synchronization Server Configuration Console main menu, enter a number corresponding to a component that you want to configure or edit.

```
>>>> UnboundID Synchronization Server configuration console main menu

What do you want to configure?

    1)  Attribute Map              8)   Log Publisher
    2)  Attribute Mapping          9)   Log Retention Policy
    3)  Connection Handler         10)  Log Rotation Policy
    4)  DN Map                     11)  Sync Class
    5)  External Server            12)  Sync Destination
    6)  Global Configuration       13)  Sync Pipe
    7)  Global Sync Configuration  14)  Sync Source

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

Enter choice:
```

Configuring Server Groups Using dsconfig Interactive

In a typical Synchronization Server deployment, administrators set up one Synchronization Server and one or more redundant failover servers. The failover servers can immediately take over from the primary server if connection is lost for any reason (see *Installing a Redundant Failover Server* on page 39 for instructions).

It is important that the primary and secondary servers have the same configuration settings to ensure the proper operation of your sync topology. To enable this, you must assign the Synchronization Servers to a server group using the `dsconfig` tool, so that any change to one server will automatically be applied to the other servers in the group.

After you have set up a server group, you can make an update on one server using `dsconfig`, then apply the change to the other servers in the group using the `--applyChangeTo server-group` option of the dsconfig non-interactive command. If you want to apply the change to one server in the group, use the `--applyChangeTo single-server` option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

To Configure Server Groups

- Run the `dsconfig` command and set the global configuration property for server groups to "all-servers". On the primary Synchronization Server, do the following:

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

  If you add redundant or failover servers to the topology, the setup tool will copy the configuration from the primary server to the new server(s).

Configuring External Servers Using dsconfig Interactive

To set up a Synchronization topology, you must define a single server of a topology of identical, replicated servers to be synchronized. For each Directory Server, you must define the host, port, SSL, bind DN, and bind password. A single external server configuration object can be referenced by multiple Sync Sources and Sync Destinations.

To Configure the External Servers Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to External Server.
2. On the External Server Management menu, type the number corresponding to create a new External Server.
3. Next, select the type of external server. In this example, select the option for Sun DS External Server.
4. Next, you will be prompted to enter the name for the external server.
5. On the Server-Host-Name Property menu, type the host name of the external server.
6. On the Sun DS External Server Properties menu, change the server-port, bind-dn, and password for the external server. Type the number corresponding to each property, and follow the prompts to enter the values. When completed, type `f` to save and apply the changes.
7. Repeat steps 2–6 to define any additional external servers. The Synchronization Server uses the settings for the first server as a template to create the other external servers. Type the number to use the first external server as a template for the other external server.
8. Repeat steps 2–7 to create the other external servers that you plan to synchronize.
9. On the External Server Management menu, type the number to view the list of external servers that you have created.

```
External Server : Type         : server-host-name : server-port
----------------:--------------:------------------:------------
ds-dest1        : UnboundID-ds : ds3.example.com  : 389
ds-dest2        : UnboundID-ds : ds4.example.com  : 389
ds-src1         : sun-ds       : ds1.example.com  : 389
ds-src2         : sun-ds       : ds2.example.com  : 389
```

Configuring the Sync Source Using dsconfig Interactive

Sync Sources define the directory topology that is the source of the data to be synchronized. When data in the Sync Source changes, it is synchronized to the Sync Destination topology. Sync Sources can reference one or more external servers of the appropriate type (UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory

Server, Alcatel-Lucent 8661 Directory Proxy Servers (3.x), Sun Directory Server 5.x, Sun DSEE 6.x, 7.x, Microsoft Active Directory, Oracle 10g,11g, or Microsoft SQL Server 2005, 2008).

To Configure the Sync Source Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to the Sync Source.
2. On the Sync Source Management menu, type the number corresponding to create a new Sync Source.
3. On the Sync Source Type menu, enter the number corresponding to the Sync Source type.
4. Next, you will be prompted to enter a name for the Sync Source. Enter a unique name for the sync source.
5. On the Base DN Property menu, enter the base DN for the Sync Source. In this example, type dc=example,dc=com, and then press **Enter** when prompted to complete the step.
6. On the Configuring the Server Property menu, select the external servers that will be part of the Sync Source topology. You can enter the number(s) corresponding to the external servers separated by commas. For example, enter "3,4" for several external servers.
7. On the Sync Source Properties menu, you can set the `ignore-changes-by-dn` property that specifies the user DN whose modifications on the external server will be ignored during synchronization. This property is useful when using the UnboundID Synchronization Server bidirectionally to limit loop back synchronization changes (modifications) back to the source by the specified user DN (for example, `uid=sync user`). Because this example is setting up a one-way sync pipe, you can type f to finish.

   The DN of the user who is performing a delete operation is not normally available in the change log. Delete operations by these users will not be ignored.

Configuring the Sync Destination Using dsconfig Interactive

Sync Destinations define the topology of directory servers where changes detected at the Sync Source are applied. Sync Destinations reference one or more external servers of the appropriate type.

To Configure the Sync Destination Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to set up the Sync Destination.
2. On the Sync Destination Management menu, type the number corresponding to creating a new Sync Destination.

3. Next, on the Sync Destination Type menu, enter the number corresponding to the Sync type (1 for UnboundID Directory Server, 2 for Microsoft Active Directory, 3 for JDBC Sync, 4 for Sun DS Sync). In this example, type the number for UnboundID Sync Destination.

4. Next, you will be prompted to enter a name for the Sync Destination. Enter a unique name for the Sync Destination.

5. On the Base DN Property menu, enter the base DN for the Sync Destination. In this example, type `dc=example,dc=com`, and then press **Enter** when prompted to complete the step.

6. On the Server Property menu, select the external servers that will be part of the Sync Destination topology. You can enter the number corresponding to the external servers separated by commas (e.g., "1,2").

7. On the Sync Destination Properties menu, type `f` to save and apply the changes.

Configuring a Sync Pipe Using dsconfig Interactive

A Sync Pipe defines a single synchronization path between the source and destination topologies. Every Sync Pipe has one or more Sync Classes that controls how and what is synchronized. Multiple Sync Pipes can run in a single UnboundID Synchronization Server instance.

Note: Once you have set up a Sync Pipe, remember to start the Sync Pipe for synchronization using the `realtime-sync start` command.

To Configure a Sync Pipe Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to the Sync Pipe.

2. On the Sync Pipe Management menu, type the number corresponding to creating a new Sync Pipe.

3. Enter a unique name for the Sync Pipe. A Sync Pipe defines a single synchronization path between the Sync Source and Sync Destination.

4. On the Sync-Source Property menu, select the Sync Source for the Sync Pipe from an existing sync source, or create a new Sync Source if it was not created in an earlier step.

5. On the Sync-Destination Property menu, select the Sync Destination for the Sync Pipe from an existing sync destination, or create a new Sync Destination if it was not created in an earlier step.

6. On the Sync Pipe Properties menu, type the number corresponding to starting the Sync Pipe, follow the prompts, and then when done, type `f` to save and apply the changes. Although the Sync Pipe has started, you must define at least one Sync Class for synchronization to work.

7. Repeat steps 1–6 to create other Sync Pipes. The Synchronization Server can have multiple Sync Pipes in the system. When done, you must define at least one Sync Class for each Sync Pipe. Within a Sync Pipe, a Sync Class defines each type of entry that needs to be treated differently.

Configuring the Sync Class Using dsconfig Interactive

Sync Classes define the operation types (e.g., creates, modifies, or deletes) and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. A source entry is in at most one Sync Class and is determined by a base DN and LDAP filters. A Sync Class can have multiple Attribute Maps and DN Maps, or none. For each Sync Pipe, a Sync Class is defined for each type of entry that needs to be treated differently.

To Configure a Sync Class for each Sync Pipe

1. On the Synchronization Server Configuration console main menu, type the number corresponding to the Sync Class.

2. On the Sync Class Management menu, type the number corresponding to creating a new Sync Class.

3. Select the Sync Pipe that will use the Sync Class. If there is only one Sync Pipe, verify that the existing Sync Pipe is the one that you are configuring, and then press **Enter** to accept the default.

4. Next, enter a name for the Sync Class that you are defining.

5. On the Sync Class Properties menu, for the Evaluation Order Index field, type the priority ordering for the Sync Class if you have more than one Sync Class configured for the topology. Sync Classes with a smaller evaluation-order-index property is evaluated first. Because this example defines only one Sync Class, the default value of 9999 is used.

6. For the Include Base DN field, type the base DN for the branches of the Sync Source that contain entries in this Sync Class. Only entries with this base DN will be included in the Sync Class. This is an optional step. If no base DN is specified, the location of the entry in the Sync Source is not taken into account when determining if an entry is part of this Sync Class.

7. For the Include Filter field, type a search filter that determines which entries are in the Sync Class. If no filter is specified, all entries within the specified included base DNs are included in the Sync Class.

8. For the Attribute Map field, enter an attribute map for the Sync Class. Because this example shows a migration path from Sun Directory Server 5.x to UnboundID Directory Server, you do not need to set up an attribute map, unless you have added new attributes to your schema. See *Configuring Attribute Maps* on page 93.

9. For the DN Map field, enter a DN map for the Sync Class. See *Configuring DN Maps*.

10. On the Sync Class Properties menu, type `f` to save and apply the changes when you have completed configuring the sync class.

11. Repeat steps 1–10 to define another Sync Class for the Sync Pipe.

Starting the Global Sync Configuration Using dsconfig Interactive

After you have set up the Synchronization topology, you must start the Global Sync Configuration, which will use only those Sync Pipes that have been started.

To Start the Global Sync Configuration

1. On the Synchronization Server Configuration console main menu, type the number corresponding to the Global Sync Configuration.

2. On the Global Sync Configuration Management menu, type the number corresponding to view and edit the configuration.

3. On the Global Sync Configuration Properties menu, type the number corresponding to setting the `started` property, and then follow the prompts to set the value to TRUE.

4. On the Global Sync Configuration Properties menu, type `f` to save and apply the changes.

Generating a Summary of Configuration Components

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

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

To Generate a Summary of Configuration Components

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

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

```
Sync Pipes:
  Sync Pipe: UnboundID Directory Server 2 to UnboundID Directory Server
    started: false
    synchronization-mode: standard
    change-detection-polling-interval: 500 ms
    num-worker-threads: 20
    sync-source:
```

```
    UnboundID Sync Source: UnboundID Directory Server 2
      base-dn: "dc=example,dc=com"
      ignore-changes-by-dn: "cn=Sync User,cn=Root DNs,cn=config"
      use-changelog-batch-request: true
      proxy-server: none
      server:
        UnboundID DS External Server: localhost:2389
          server-host-name: localhost
          server-port: 2389
          bind-dn: "cn=Sync User,cn=Root DNs,cn=config"
          password: ********
          connection-security: none
          authentication-method: simple
          allowed-operation: abandon, add, bind, compare, delete, extended,
           modify, modify-dn, search
          trust-manager-provider: none
          key-manager-provider: none
  sync-destination:
    UnboundID Sync Destination: UnboundID Directory Server
      base-dn: "dc=example,dc=com"
      server:
        UnboundID DS External Server: localhost:1389
          server-host-name: localhost
          server-port: 1389
          bind-dn: "cn=Sync User,cn=Root DNs,cn=config"
          password: ********
          connection-security: none
          authentication-method: simple
          allowed-operation: abandon, add, bind, compare, delete, extended,
           modify, modify-dn, search
          trust-manager-provider: none
          key-manager-provider: none
      proxy-server: none
  Sync Classes:
    Sync Class: test sync class 2
      evaluation-order-index: 10
      include-base-dn: "ou=sites,dc=example,dc=com"
      include-filter: (objectClass=site), (siteName=u*)
      auto-mapped-source-attribute: -all-
      excluded-auto-mapped-source-attributes: No source attributes are excluded
       from synchronization.
      destination-correlation-attributes: dn
      synchronize-creates: true
      synchronize-modifies: true
      synchronize-deletes: true
      allow-destination-renames: true
      dn-map: none
      attribute-map: none
    Sync Class: DEFAULT
      evaluation-order-index: 9999
      include-base-dn: The location of the entry in the Sync Source is not taken
       into account when determining whether an entry is part of this Sync Class.
      include-filter: All entries are included in this Sync Class.
      auto-mapped-source-attribute: -all-
      excluded-auto-mapped-source-attributes: No source attributes are excluded
       from synchronization.
      destination-correlation-attributes: dn
      synchronize-creates: false
      synchronize-modifies: false
      synchronize-deletes: false
      allow-destination-renames: true
      dn-map: none
      attribute-map: none

  Sync Pipe: UnboundID Directory Server to UnboundID Directory Server 2
    started: false
    synchronization-mode: standard
    change-detection-polling-interval: 500 ms
    num-worker-threads: 20
    sync-source:
      UnboundID Sync Source: UnboundID Directory Server
        base-dn: "dc=example,dc=com"
    ignore-changes-by-dn: "cn=Sync User,cn=Root DNs,cn=config"
    use-changelog-batch-request: false
    proxy-server: none
    server:
      UnboundID DS External Server: localhost:1389
        server-host-name: localhost
        server-port: 1389
        bind-dn: "cn=Sync User,cn=Root DNs,cn=config"
```

```
          password: ********
          connection-security: none
          authentication-method: simple
          allowed-operation: abandon, add, bind, compare, delete, extended,
           modify, modify-dn, search
          trust-manager-provider: none
          key-manager-provider: none
    sync-destination:
      UnboundID Sync Destination: UnboundID Directory Server 2
        base-dn: "dc=example,dc=com"
        server:
          UnboundID DS External Server: localhost:2389
            server-host-name: localhost
            server-port: 2389
            bind-dn: "cn=Sync User,cn=Root DNs,cn=config"
            password: ********
            connection-security: none
            authentication-method: simple
            allowed-operation: abandon, add, bind, compare, delete, extended,
             modify, modify-dn, search
            trust-manager-provider: none
            key-manager-provider: none
        proxy-server: none
    Sync Classes:
      Sync Class: test sync class
        evaluation-order-index: 10
        include-base-dn: "ou=people,dc=example,dc=com"
        include-filter: (uid=user.*)
        auto-mapped-source-attribute: description, email, password
        excluded-auto-mapped-source-attributes: No source attributes are excluded
         from synchronization.
        destination-correlation-attributes: dn
        synchronize-creates: true
        synchronize-modifies: true
        synchronize-deletes: true
        allow-destination-renames: true
        dn-map:
          DN Map: test dn map
            from-dn-pattern: "*,**,dc=com"
            to-dn-pattern: "uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}
             {eid},{2},o=example"
        attribute-map:
          Attribute Map: test attribute map
            Attribute Mappings:
              Direct Attribute Mapping: username
                to-attribute: username
                from-attribute: uid
              Constructed Attribute Mapping: description
                to-attribute: description
                value-pattern: {givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid}
              DN Attribute Mapping: email
                to-attribute: email
                from-attribute: firstname
                dn-map:
                  DN Map: test dn map
                    from-dn-pattern: "*,**,dc=com"
                    to-dn-pattern: "uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}
                     {eid},{2},o=example"
    Sync Class: DEFAULT
      evaluation-order-index: 9999
      include-base-dn: The location of the entry in the Sync Source is not taken
       into account when determining whether an entry is part of this Sync Class.
      include-filter: All entries are included in this Sync Class.
      auto-mapped-source-attribute: -all-
      excluded-auto-mapped-source-attributes: No source attributes are excluded
       from synchronization.
      destination-correlation-attributes: dn
      synchronize-creates: true
      synchronize-modifies: true
      synchronize-deletes: true
      allow-destination-renames: true
      dn-map: none
      attribute-map: none


Global Sync Configuration:
  started: true
  changelog-password-decryption-key: -
  sync-failover-polling-interval: 7500
```

Preparing the Synchronization Server for External Server Communication

The UnboundID® Synchronization Server provides a tool, `prepare-endpoint-server`, that sets up any communication variances that may occur between the Synchronization Server and the external servers. Typically, directory servers can have different security settings, privileges, and passwords (e.g., for trust stores) configured on the Sync Source that would reject any import of entries in the Sync Destination.

The `prepare-endpoint-server` tool also creates a Synchronization User Account and its privileges on all of the external servers (see *About the Sync User Account* for more detailed information). If necessary, you will be prompted for the root or administrator credentials (for example, `uid=admin,dc=example,dc=com`) to set up this user account. The `prepare-endpoint-server` tool also checks if the sync-user account has the proper privileges to access the `firstChangeNumber` and `lastChangeNumber` attributes in the root DSE entry so that it can get the most up-to-date changes to the system. If the Sync User does not have the proper privileges, the Synchronization Server displays a warning message. You can view any warning or error messages in the `logs/prepare-endpoint-server.log` file.

Note: If you created your Synchronization topology using the `create-sync-pipe-config` tool, then you do not need to run this command separately as it is already part of the process.

To Prepare the Synchronization Server for External Server Communication

1. Use the `prepare-endpoint-server` tool to prepare the directory server instances on the remote host for synchronization as a data source for the subtree, `dc=example,dc=com`. If the user account is not present on the external server, then the Synchronization Server will create it if it has a parent entry.

```
$ bin/prepare-endpoint-server \
  --hostname sun-ds1.example.com --port 21389 \
  --syncServerBindDN "cn=Sync User,dc=example,dc=com" \
  --syncServerBindPassword secret --baseDN "dc=example,dc=com" \
  --isSource
```

2. When prompted, enter the bind DN and password to create the user account. This step enables the change log database and sets the `changelog-maximum-age` property to some recommended value.

3. Repeat steps 1–2 for the other external source servers. Remember to specify the host name and port number of the external server.

4. For the destination servers, repeat steps 2–3 but remember to include the `--isDestination` option. If your destination servers do not have any entries, then a "Denied" message will be generated when creating the cn=Sync User entry as no base DN exists.

```
$ bin/prepare-endpoint-server \
  --hostname UnboundID-ds1.example.com --port 33389 \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword sync --baseDN "dc=example,dc=com" \
  --isDestination
```

5. Repeat step 4 for the other Destination servers.

Preparing External Servers: If the Admin Does Not Have Root Access on DSEE External Servers

If you are syncing from a Sun DSEE external endpoint server and do not have root access to those machines, you can provide the following manual steps to someone who does have root access on the machines.

To Set Up the DSEE External Servers

1. Complete the Synchronization Server configuration using the `create-sync-pipe-config` tool or the `dsconfig` command.

2. Make sure that the Sync User account is created outside of the `cn=config` branch. We have seen problems with DSEE when the Sync User is placed there. If the Sync User is in `cn=config`, delete it, add it to the normal backend (e.g., dc=example,dc=com), and then update the configuration in the Synchronization Server. For example, create an LDIF file, and save it as "sync-user.ldif". When configuring DSEE endpoint servers, you must include resource limit attributes in the `cn=Sync User` entry, so that `resync` can conduct searches throughout the whole directory. The `nsLookThroughLimit` operational attributes determines the maximum number of entries checked during a search. The `nsTimeLimit` operational attribute determines the maximum time spent processing a search operation. The `nsIdleTimeout` operational attribute determines the maximum amount of time that a client connection can remain idle before it is dropped. The `nsSizeLimit` operational attribute determines the maximum number of returned entries for a search operation. All of these attributes are set to `-1`, which means that there is no limit for each respective parameter.

```
dn: cn=Sync User,dc=example,dc=com
cn: Sync User
givenName: Sync
sn: User
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
userPassword: password
nsLookThroughLimit: -1
nsTimeLimit: -1
nsIdleTimeout: -1
nsSizeLimit: -1
```

3. Add the following entry to the external DSEE server:

```
$ ldapmodify -h {dsee-host} -p {ldap-port} -D "cn=directory manager" -w {password} \
  -a -f sync-user.ldif
```

4. On the DSEE server, perform a search to see what arguments are already set on the Retro Change Log plug-in. Look for arguments for the attribute, `nsslapd-pluginarg[0-9]`. In the following example, we see that `nsslapd-pluginarg0` and `nsslapd-pluginarg1` are already present, so we need to use `nsslapd-pluginarg2` for any additional settings.

```
$ ldapsearch -h {dsee-host} -p {ldap-port} -D "cn=directory manager" -w {password} \
  -b "cn=Retro Changelog Plugin,cn=plugins,cn=config" -s base "(objectclass=*)"

dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: ds-signedPlugin
objectClass: extensibleObject
cn: Retro Changelog Plugin
nsslapd-pluginPath: /ds/upc/servers/sunds52/lib/retrocl-plugin.so
nsslapd-pluginInitfunc: retrocl_plugin_init
nsslapd-pluginType: object
nsslapd-plugin-depends-on-type: database
nsslapd-changelogdir: /ds/upc/servers/sunds52/slapd-upc/db/changelog
nsslapd-pluginEnabled: on
nsslapd-changelogmaxage: 3d
nsslapd-pluginarg0: -ignore_attributes
nsslapd-pluginarg1: copyingFrom
nsslapd-pluginId: retrocl
nsslapd-pluginVersion: 5.2_Patch_4
nsslapd-pluginVendor: Sun Microsystems, Inc.
nsslapd-pluginDescription: Retrocl Plugin
ds-pluginSignatureState: valid signature
```

5. On the DSEE server, enable the Retro Change Log Plug-in using the console or command-line tool. Use `ldapmodify` to apply the following LDIF to the server, or you can make the equivalent changes to `dse.ldif` after the server has been shutdown. The LDIF file enables the Retro Change Log plug-in, sets the max age to three days, and adds the `deletedEntryAttrs` setting into one of the `nsslapd-pluginarg` fields (see below). The `deletedEntryAttrs` attribute is used to ensure that the Synchronization Server has the proper information for the correlation of deletes against the target system. The attribute will be used to record `objectclass`, `cn`, `uid`, and `modifiersName` during deletes. You can modify this list of attributes so that the Synchronization Server can find the corresponding entry in the destination server. In this example, make sure to use the `nsslapd-pluginarg2` attribute name to add the `deletedEntryAttrs` parameters as `nsslapd-pluginarg0` and `nsslapd-pluginarg1` are in use. Finally, save the file as `retro-changelog-enable.ldif`.

```
dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
-
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 3d
-
replace: nsslapd-pluginarg2
nsslapd-pluginarg2: deletedEntryAttrs=objectclass,cn,uid,modifiersName
```

6. Use `ldapmodify` to enable the Retro Change Log Plug-in.

```
$ ldapmodify -h {dsee-host} -p {ldap-port} -D "cn=directory manager" -w {password} \
  -f retro-changelog-enable.ldif
```

7. Restart DSEE so that the plug-in can start recording changes.

8. Create an LDIF file called `sync-dsee-aci.ldif` to add an ACI so that the Sync User can access the change log and data, respectively.

```
dn: cn=changelog
```

```
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "UnboundID Sync User Access"; allow
 (read,search,compare) userdn="ldap:///cn=Sync User,dc=example,dc=com";)

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "UnboundID Sync User Read/Write Access";
 allow (all) userdn="ldap:///cn=Sync User,dc=example,dc=com";)
```

If the DSEE server is only used as a source, and no modifies will be performed against the server, then the ACI should be as follows:

```
dn: dc=changelog
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "UnboundID Sync User Access"; allow
 (read,search,compare) userdn="ldap:///cn=Sync User,dc=example,dc=com";)

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "UnboundID Sync User Read Only Access"; allow
 (read,search,compare) userdn="ldap:///cn=Sync User,dc=example,dc=com";)
```

9. Use `ldapmodify` to add the `sync-dsee-aci.ldif` to the DSEE server.

```
$ ldapmodify -h {dsee-host} -p {ldap-port} -D "cn=directory manager" -w {password} \
  -f sync-dsee-aci.ldif
```

Using Resync on the Synchronization Server

The UnboundID® Synchronization Server provides a bulk synchronization command-line tool, resync, that can be used to verify the Synchronization setup. The `resync` tool operates on a single Sync Pipe at a time and retrieves entries from the Sync Source in bulk, and compares the source entries with the corresponding destination entries. If destination entries are missing or attributes out-of-sync, then the Synchronization Server updates them.

The command provides a `--dry-run` option that can be run to test the matches between Sync Source and Destination but not commit any changes to the target topology. The resync tool also provides options to write debugging output to a log with a configurable level of verbosity for testing purposes.

Note: While you can use the `resync` tool to update any mismatched entries, you should use the tool only for relatively small datasets. For large deployments, you can export entries from the Sync Source into an LDIF file, run the `translate-ldif` tool to translate and filter the entries into the destination format, and then import the result LDIF file into the Sync Destination.

Typically, you can use the `resync` tool to verify the synchronization configuration after it has been configured. The command has some important options that can be used to test the configuration:

Table 8: Useful Resync Command Options

Resync Option	Description
--dry-run	Reports the sync status of the configuration without committing the change to the target topology.
--numPasses	Specifies the number of passes to compare an entry that is out-of-sync to account for synchronization delays. If both Sync Source and Sync Destination are quiescent, then a value of 1 can be provided.
--logFilePath	Specifies the path to the log file that records the details of the resync operation.
--logLevel	Specifies the resync log level that controls the amount of logging. The following levels are available: • **out-of-sync-summary**. Provides a single summary message for each missing or out-of-sync entry • **out-of-sync-detailed**. Provides a single detailed message including the source and destination entry contents. • **all-entries-summary**. Provides multiple summary messages, which are logged for every entry that is loaded or compared. The contents of the entries are not included. • **all-entries-detailed**. Provides multiple detailed messages, which are logged for every entry that is loaded or compared. This option can impact performance as it generates a large output file. • **debug**. Provides multiple verbose messages, which are logged for every entry that is loaded or compared. This option should only be used to diagnose or troubleshoot a problem as its potential size could impact performance. The contents of the entries are included during processing.
--ratePerSecondFile	Specifies a specific synchronization rate (synchronizing changes per second). The option allows you to adjust the rate during off-peak hours or adjust the rate based on measured loads for very long running resync operations. The file must contain a single positive integer number surrounded by white space (for example, 1) to start with. If the file is updated with an invalid number (for example, changing it to zero, a negative number, or something other than an integer number), the rate is not updated. To use this feature, run resync first at 100 operations/sec, measure the impact on the source servers, then adjust as desired.
--secondsBetweenPass	Specifies the number of seconds to wait between each pass to recheck entries that were out-of-sync. This option is used when entries are out-of-sync due to synchronization delays.
--sourceInputFile	Specifies a file containing a list of DNs to be retrieved from the Sync Source and processed. The option allows for faster processing of very large data sets by targeting individual base-level searches for each source DN in the file. For LDAP Sync Sources, this file should contain a list of DNs; for JDBC Sync Sources, the data may be in a user-defined format since it will be consumed by a JDBC Sync Source extension. When synchronizing with a database, you can use the `--entryType` option that specifies the type of database entry to search for. This must match one of the configured entry types in the JDBCSyncSource

The `resync` tool provides a number of other useful functions, including the ability to schedule a nightly synchronization if real-time synchronization is not necessary (for example, the creation of new entries during a specific time period can be resynced at a designated nightly time). The tool also provides explicit control over which attributes are included or excluded during the synchronization process if fine-grained synchronization is required by the Attribute or DN maps. For more information, type `bin/resync --help` for information and examples.

Testing Attribute and DN Maps Using Resync

You can use the `resync` tool to test how attribute maps and DN maps are configured by synchronizing a single entry. If the `--logFilePath` and `--logLevel` options are specified, the resync tool generates a log file with varying degrees of details to show any synchronization messages. You can specify the log file and the level of detail of processing messages.

To Test Attribute and DN Maps Using Resync

- Use the `resync` tool in "dry run" mode by specifying a single entry. Assume that the Sync Source topology contains an entry, uid=user.0. Any logging performed during a resync operation appears in the `logs/tools/resync.log`.

```
$ bin/resync --pipe-name sun-to-UnboundID-sync-pipe \
  --sourceSearchFilter "(uid=user.0)" --dry-run --logLevel debug
```

Verifying the Synchronization Configuration Using Resync

The most common example for resync is to test that the Sync Pipe configuration has been set up correctly. For example, the following procedure assumes that the configuration was set up with the Sync Source topology (two replicated Sun Directory Server 5.x servers) with 2003 entries; the Sync Destination topology (two replicated UnboundID Directory Servers) has only the base entry and the cn=Sync User entry. Both Source and Destination topologies have their LDAP Change Logs enabled. Because both topologies are not actively being updated, the resync tool can be run with one pass through the entries.

To Verify the Synchronization Configuration Using Resync

Use `resync` with the `--dry-run` option to check the synchronization configuration. The following example does a dry-run process to verify the Sync configuration and creates entries that are not present in the Source Destination. The output also displays a timestamp that can be tracked in the logs.

```
$ bin/resync --pipe-name sun-to-UnboundID-sync-pipe --numPasses 1 --dry-run

Starting Pass 1

Status after completing all passes[20/Mar/2010:10:20:07 -0500]
---------------------------------
Source entries retrieved      2003
Entries missing               2002
Entries out-of-sync           1
Duration (seconds)            4

Resync completed in 4 s.

0 entries were in-sync, 0 entries were modified, 0 entries were created,
1 entries are still out-of-sync, 2002 entries are still missing, and
0 entries could not be processed due to an error
```

Populating an Empty Sync Destination Topology Using Resync

The `resync` tool can populate an empty Sync Destination with the Sync Source entries prior to real-time synchronization. If you already have data from the Sync Source in the Sync Destination, you can use the resync tool to synchronize entries with the Sync Source.

The following procedures shows how you can use resync to populate an empty Sync Destination topology for small datasets. For large deployments, see *To Populate an Empty Sync Destination Topology Using translate-ldif*.

To Populate an Empty Sync Destination Topology Using Resync

1. In this example, assume that the Sync Destination topology has only the base entry (dc=example,dc=com) and the cn=Sync User entry. Run `resync` in a dry-run (see the previous example). Assume an error was generated during the process.

2. Rerun the `resync` command with the log file path and with the log level debug. Do not include the `--dry-run` option. Any logging performed during a resync operation appears in the `logs/tools/resync.log`.

```
$ bin/resync --pipe-name sun-to-UnboundID-sync-pipe \
  --numPasses 1 --logLevel debug
```

3. Open the `logs/resync-failed-DNs.log` file in a text editor to locate the error and fix it. As seen below, sometimes an entry cannot be created because the parent entry does not exist. After creating the parent entry on the destination (`ou=People,dc=example,dc=com`), you can rerun the `resync` command to create the missing entries.

```
# Entry '(see below)' was dropped because there was a failure at the resource:
Failed to create entry uid=mlott,ou=People,dc=example,dc=com.  Cause:
LDAPException(resultCode=no such object, errorMessage='Entry
uid=user.38,ou=People,dc=example,dc=com cannot be added because its parent
entry ou=People,dc=example,dc=com does not exist in the server',
 matchedDN='dc=example,dc=com')
(id=1893859385ResourceOperationFailedException.java:126 Build revision=4881)
dn: uid=user.38,ou=People,dc=example,dc=com
```

4. Rerun the `resync` command. The command creates the entries in the Sync Destination topology that are present in the Sync Source topology.

```
$ bin/resync --pipe-name sun-to-UnboundID-sync-pipe
```

```
...(output from each pass)...

Status after completing all passes[20/Mar/2010:10:23:33 -0500]
----------------------------
Source entries retrieved  160
Entries in-sync           156
Entries created           4
Duration (seconds)        11

Resync completed in 12s.

156 entries were in-sync, 0 entries were modified, 4 entries were created, 0 entries
are still out-of-sync, 0 entries are still missing, and 0 entries could not be
processed due to an error
```

Populating an Empty Sync Destination Topology Using translate-ldif

If you populate a Sync Destination using the resync tool, it could take some time to load a large dataset. For a faster method, you can use the `translate-ldif` tool to populate an empty Sync Destination topology for a very large number of entries. The `translate-ldif` tool translates the contents of an LDIF file in Sync Source format to Sync Destination format using the filtering and mapping criteria defined for the Sync Pipe's Sync Classes.

To Populate an Empty Sync Destination Topology Using translate-ldif

1. On a Sync Source Server, export the data to an LDIF file.
2. On the Synchronization Server, run the `translate-ldif` tool to translate or filter the entries into the Sync Destination format, if necessary. Make sure to specify the path to the LDIF file on the Sync Source server and the path to the output file.

```
$ bin/translate-ldif --pipe-name sun-to-UnboundID-sync-pipe \
  --sourceLDIF /path/to/sync-source-data.ldif \
  --destinationLDIF /path/to/sync-dest-data.ldif
```

3. On a Sync Destination Server, import the data using the path to the translated LDIF file.

Setting the Synchronization Rate Using Resync

The `resync` command has a `--ratePerSecondFile` option that allows you to set a specific synchronization rate (sync changes per second). The option allows you to adjust the rate during off-peak hours or adjust the rate based on measured loads for very long running resync operations by simply changing the rate in the file.

To use this feature, run `resync` first at 100 operations/sec, measure the impact on the source servers, then adjust as desired. The file must contain a single positive integer number surrounded by white space (for example, 1) to start with. If the file is updated with an invalid number (for example, changing it to zero, a negative number, or something other than an integer number), the rate is not updated.

To Set the Synchronization Rate Using Resync

1. Create a text file containing the resync rate. The number must be a positive integer surrounded by white space.

```
$ echo '100 ' > rate.txt
```

2. Run the `resync` command with the `--ratePerSecondFile` option.

```
$ bin/resync --pipe-name "sun-to-UnboundID-sync-pipe" \
  --ratePerSecondPath rate.txt
```

Note: The `resync` command also has a `--ratePerSecond` option that allows you to set the sync rates per second by specifying the target rate. The option allows you to throttle resync and reduce its load on the end servers. If this option is not provided, then the tool resyncs at the maximum rate.

3. Check the rate on your system, and then update the rate file again to change the resync rate.

```
$ echo '150 ' > rate.txt
```

Synchronizing a Specific List of DNs

The `resync` command allows you to synchronize a specific set of DNs that are read from a file using the `--sourceInputFile` option. The option is most useful for very large datasets that require faster processing by targeting individual base-level searches for each source DN in the file. If any DN fails for any reason (parsing, search, or process errors), the command creates an output file of the skipped entries (`resync-failed-DNs.log`), which can be rerun again.

The file must contain only a list of DNs in LDIF format with "dn:" or "dn::". The file can include comment lines by starting each line with a pound sign (#). All DNs can be wrapped and are assumed to be wrapped on any lines that begin with a space followed by text. Empty lines are ignored.

For small files, you can create a file manually. For large files, you can use ldapsearch to create an LDIF file, as seen below.

To Synchronize a Specific List of DNs

1. To create a file of DNs, you can enter each manually for small files, or you can run an `ldapsearch` command using the special OID "1.1" extension, which only returns the DNs in your DIT. For example, on the Sync Source directory server, run the following command:

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com \
  --baseDN dc=example,dc=com --searchScope sub "(objectclass=*)" "1.1" > dn.ldif
```

2. Task step.

```
$ bin/resync --pipe-name "sun-to-UnboundID-pipe" --sourceInputFile dn.ldif

Starting pass 1

[20/Mar/2010:10:32:11 -0500]
------------------------------------
Resync pass                     1
Source entries retrieved        1999
Entries created                 981
Current pass, entries processed 981
Duration (seconds)              10
Average ops/second              98

Status after completing all passes[20/Mar/2010:10:32:18 -0500]
---------------------------------
Source entries retrieved      2003
Entries created               2003
```

```
Duration (seconds)            16
Average ops/second            98
Resync completed in 16 s.

0 entries were in-sync, 0 entries were modified, 2003 entries were created, 0 entries
are still out-of-sync, 0 entries are still missing, and 0 entries could not be
processed due to an error
```

3. If any errors occurred, view the `logs/tools/resync-failed-DNs.log` to see the skipped DNs. Then, correct the source DNs file, and rerun the `resync` command.

Controlling Real Time Synchronization

In real-time mode, the UnboundID Synchronization Server polls the source server for changes and synchronizes the destination entries immediately. Once the UnboundID Synchronization Server determines that a detected change should be included in the synchronization, it fetches the full entry from the source. Then, it finds the corresponding entry in the destination end-point using flexible correlation rules and applies the minimum set of changes to bring the attributes that were modified into sync. The server fetches and compares the full entries to make sure it does not synchronize any stale data from the change log.

About the Realtime-Sync Tool

The UnboundID Synchronization Server provides a utility to control real-time synchronization including starting and stopping synchronization globally or for individual Sync Pipes. The tool also provides features to set a specific starting point for real-time synchronization, so that changes made before the current time are ignored, and to schedule a stop or start at a future date.

Table 9: Realtime-Sync Command Options

Realtime-Sync Options	Descriptions
start	Start synchronization globally or for a specific Sync Pipe.
stop	Stop synchronization globally or for a specific Sync Pipe.
set-startpoint	Start synchronization for a specific Sync Pipe at a specified time. When specified, all changes made prior to the current time the command is invoked will be ignored by the Sync Pipe. Additional options include: • **--change-number {change number}**. Begin synchronization at a specific change number in the change log. This feature cannot be used if the endpoint server is the UnboundID Directory Proxy Server. See "Syncing Through Proxy Servers" for more information. • **--startpoint-rewind {duration}**. Begin synchronization by "rewinding" or starting the synchronization back at a specified duration from the current time. The duration string has the format: d (days), h (hours), m (minutes), s (seconds), ms (milliseconds). For example, to start the synchronization state that occurred 1 day, 2 hours, 12 minutes, and 30 seconds, use "1d2h12m30s". You can also specify milliseconds, for example, "300ms". The `set-startpoint` option cannot be run on a Sync Pipe that has already started.

Note: To get an accurate picture of the current status of real-time synchronization, view the monitor properties: `num-sync-ops-in-flight`, `num-ops-in-queue`, and `source-unretrieved-changes`. For example, use `ldapsearch` to view a specific Sync Pipe's monitor information:

```
$ bin/ldapsearch --baseDN cn=monitor --searchScope sub "(cn=Sync Pipe
 Monitor: PIPE_NAME)"
```

Another useful tool is the Periodic Stats Logger.

Starting Real Time Synchronization Globally

You can start real time synchronization globally for all Sync Pipes using the `realtime-sync` tool in the `bin` directory (or `bat` directory for Microsoft Windows systems). The command assumes that you have properly configured your Synchronization topology.

To Start Real Time Synchronization Globally

1. Use `realtime-sync` to start a synchronization topology globally. Assume that a single Sync Pipe called "dsee-to-UnboundID-sync-pipe" exists.

   ```
   $ bin/realtime-sync start --pipe-name "dsee-to-UnboundID-sync-pipe" \
     --port 389 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
   ```

2. If you have more than one Sync Pipe configured, specify each Sync Pipe using the `--pipe-name` option. The following example starts `realtime-sync` for a bidirectional synchronization topology.

   ```
   $ bin/realtime-sync start --pipe-name "Sun DS to UnboundID DS" \
     --pipe-name "UnboundID DS to Sun DS" --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
   ```

Pausing Synchronization

You can pause or start synchronization by using the 'start' and 'stop' subcommands. If synchronization is stopped and then restarted, then changes made at the Sync Source while synchronization was stopped will still be detected and applied.

Synchronization for individual Sync Pipes can be started or stopped using the `--pipe-name` argument. If the `--pipe-name` argument is omitted, then synchronization is started or stopped globally.

To Stop Real Time Synchronization Globally

- Use `realtime-sync` to stop a synchronization topology globally. This command will stop all Sync Pipes started.

  ```
  $ bin/realtime-sync stop --port 389 --bindDN "uid=admin,dc=example,dc=com" \
    --bindPassword secret --no-prompt
  ```

To Stop an Individual Sync Pipe

- Use `realtime-sync` to stop an individual Sync Pipe. Assume the topology has two Sync Pipes, Sync Pipe1 and Sync Pipe2. This command stops Sync Pipe1.

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
```

Setting Startpoints

You can set startpoints that instructs the Sync Pipe to ignore all changes made prior to the current time using the set-startpoint subcommand with the `realtime-sync` command. Once synchronization is started, only changes made after this command is run will be detected at the Sync Source and applied at the Sync Destination.

The `set-startpoint` subcommand is often run during the initial setup prior to starting real-time synchronization for the first time. It should be run prior to initializing the data in the Sync Destination, which is usually done either by using the resync command or by exporting data from the Sync Source, running `translate-ldif`, and then importing the data into the Sync Destination.

The `set-startpoint` subcommand also has two convenient options that can start synchronization at a specific change log number or back at a sync state that occurred at a specific time duration ago (for example, you can start synchronizing at a sync state that occurred 10 minutes ago from the current time).

To Set a Synchronization Startpoint

1. Stop the synchronization topology globally (if it had been started previously) using the `realtime-sync` command with the stop subcommand.

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

2. Set the startpoint for the synchronization topology. Any changes made before setting this command will be ignored.

```
$ bin/realtime-sync set-startpoint --pipe-name "Sync Pipe1" \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

```
Set StartPoint task 2011072109564107 scheduled to start immediately
[21/Jul/2011:09:56:41 -0500] severity="INFORMATION" msgCount=0 msgID=1889535170
message="The startpoint has been set for Sync Pipe 'sun-to-UnboundID-sync-pipe'.
Synchronization will resume from the last change number in the Sync Source"
Set StartPoint task 2011072109564107 has been successfully completed
```

To Restart the Sync at a Specific Change Log Event

1. First, search for a specific change log event from which you want to restart the synchronization state. On one of the endpoint servers, run `ldapsearch` to search the change log.

```
$ bin/ldapsearch -p 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=changelog --dontWrap
  "(objectclass=*)"
```

```
  dn: cn=changelog
  objectClass: top
  objectClass: untypedObject
  cn: changelog

  dn: changeNumber=1,cn=changelog
  objectClass: changeLogEntry
  objectClass: top
  targetDN: uid=user.13,ou=People,dc=example,dc=com
  changeType: modify
  changes::
  cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwMTM4Ci0KcmVwbGFjZTogbW9kaW
  ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
  IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
  ltZTo6IEFBQUJKZ25OWlUwPQotCgA=
  changenumber: 1
  ... (more output)
  dn: changeNumber=2329,cn=changelog
  objectClass: changeLogEntry
  objectClass: top
  targetDN: uid=user.49,ou=People,dc=example,dc=com
  changeType: modify
  changes::
  cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwNDMzCi0KcmVwbGFjZTogbW9kaW
  ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
  IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
  ltZTo6IEFBQUJKZ25OMC84PQotCgA=
  changenumber: 2329
```

2. Restart synchronization from change number 2329 using the `realtime-sync` tool. Any event before this change number will not be synchronized to the target endpoint.

```
$ bin/realtime-sync set-startpoint --change-number 2329 \
  --pipe-name "Sync Pipe 1" --bindPassword secret --no-prompt
```

To Rewind the Sync State by a Specific Time Duration

The following command will start begin synchronizing data at the state that occurred 2 hours and 30 minutes ago from the current time on External Server 1 for Sync Pipe "Sync Pipe 1". Any changes made before this time will not be synchronized to the target servers. You can specify days (d), hours (h), minutes (m), seconds (s), or milliseconds (ms).

- Use `realtime-sync` with the --startpoint-rewind option to "rewind" the synchronization state and begin synchronizing at the specified time duration ago.

```
$ bin/realtime-sync set-startpoint --startpoint-rewind 2h30m \
  --pipe-name "Sync Pipe 1" --bindPassword secret --no-prompt
```

Scheduling a Realtime Sync as a Task

The `realtime-sync` tool features both an offline mode of operation as well as the ability to schedule an operation to run within the Synchronization Server's process. To schedule an operation, supply LDAP connection options that allow this tool to communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later time. Once scheduled, tasks can be managed using the `manage-tasks` tool.

To Schedule a Realtime Sync as a Task

1. Use the `--start` option with the `realtime-sync` command to schedule a start for the synchronization topology. The following command will set the start time at July 21, 2009 at 12:01:00 AM. You can also schedule a stop using the `stop` subcommand.

```
$ bin/realtime-sync set-startpoint \
  --pipe-name "sun-to-UnboundID-sync-pipe" \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --start 20110721000100 --no-prompt

Set StartPoint task 2009072016103807 scheduled to start Jul 21, 2011 12:01:00 AM CDT
```

2. Run the `manage-tasks` tool to manage or cancel the schedule task.

```
$ bin/manage-tasks --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
```

Configuring Attribute Maps

Attribute Maps are collections of Attribute Mappings, where each mapping defines those destination attributes and value that differ from that of source attributes and how the system will translate the data from one system to another. There are three types of Attribute mappings that can be defined:

- **Direct mapping**. Attributes are directly mapped to another attribute. For example, employeenumber->employeeid
- **Constructed Mapping**. Destination attribute values are derived from source attribute values and static text. For example: {givenname}.{sn}@example.com->mail
- **DN Mapping**. Attributes are mapped for attributes that store DNs. You can reference the same DN maps that map entry DNs. For example, an attribute called `manager`.

The Synchronization Server automatically validates any attribute mapping prior to applying the configuration.

Configuring an Attribute Map Using dsconfig Interactive

You can use the `dsconfig` tool in interactive mode to create an attribute map. A Sync Class can reference multiple Attribute Maps. Multiple Sync Classes can share the same Attribute Map.

To Configure an Attribute Map Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to the Attribute Map management menu.
2. On the Attribute Map management menu, type the number corresponding to creating a new attribute map.
3. Next, enter a name for the Attribute Map.
4. On the Attribute Map Property menu, type the number corresponding to entering a general description for the Attribute Map. This step is optional. Follow the prompts to enter a description for the Attribute Map. When completed, type `f` to save the changes and apply.

   You have successfully created an attribute map. Next, you must create specific add attribute mappings to your map.

Note: You can use `dsconfig` in non-interactive mode to create an attribute for easy scripting.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name test-attribute-map \
  --set "description:Test Attribute Map" \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configuring an Attribute Mapping Using dsconfig Interactive

You can use the `dsconfig` tool in non-interactive mode to create one or more attribute mappings. In this example, the Attribute Mapping sets up a Direct Mapping for one attribute: `employeeNumber` -> `employeeID`.

Note: The Synchronization Server provides a `scramble-value` advanced property that can be configured with each Attribute Mapping. The scramble feature allows you to load a Sync Destination topology with the scrambled values of real production data attributes. Obfuscating production data is convenient in testing environments.

To Configure an Attribute Mapping Using dsconfig Interactice

1. On the Synchronization Server Configuration console main menu, type the number corresponding to displaying the Attribute Mapping Management menu.

2. On the Attribute Mapping Management menu, type the number corresponding to creating a new mapping.

3. Select the attribute map that you want to configure. If there is only one Attribute Map, press **Enter** to accept the default.

4. Select the type of Attribute Mapping that you want to create: 1 for Constructed, 2 for Direct Attribute, 3 for DN Attribute. In this example, type 2 for a Direct Attribute Mapping.

5. Enter a name for the `to-attribute` for the Direct Attribute Mapping. For this example, type `employeeID`.

6. Enter a name for the `from-attribute` for the Direct Mapping. For this example, type `employeeNumber`.

7. On the Directory Attribute Mapping menu, type `f` to save and apply the changes.

8. After you have configured your Attribute Mappings, remember to add the new Attribute Map to a new Sync Class or modify an existing Sync Class.

Configuring an Attribute Mapping Using dsconfig Non-Interactive

You can use the `dsconfig` tool in non-interactive mode to create an attribute mapping. You can view the log of all configuration changes in the `logs/config-audit.log` as well as view the analogous commands to back out of each change.

To Configure an Attribute Mapping Using dsconfig Non-Interactive

1. On the Synchronization Server, use `dsconfig` in non-interactive mode to create an Attribute Mapping.

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name test-attribute-map \
  --mapping-name employeeID \
  --type direct \
  --set from-attribute:employeeNumber \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. After you have configured your Attribute Mappings, remember to add the new Attribute Map to a new Sync Class or modify an existing Sync Class.

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name test-sync-pipe \
  --class-name test-sync-class \
  --set attribute-map:test-attribute-map \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
```

```
  --bindPassword secret
```

Configuring the Directory Server Backend for Synchronizing Deletes

One important attribute that must be configured on the directory server's change log backend is the `changelog-deleted-entry-include-attribute` property. The property specifies which attributes should be recorded in the change log entry during a DELETE operation. Normally, the Synchronization Server cannot correlate a deleted entry to the entry on the destination as there is not enough information to figure out what was deleted. If you have a Sync Class configured with a filter, such as "`include-filter: objectClass=person`," then you need the `objectClass` attribute to be recorded in the change log entry. Likewise, if you have special correlation attributes (other than DN), you will need those attributes recorded on the change log entry to be properly synchronized at the endpoint server.

To Configure the Changelog-Deleted-Entry-Include-Attribute Property

- On each directory server backend (UnboundID Directory Server), use the `dsconfig` command to set the property. Remember to add the connection parameters specific to your server (hostname, port, bind DN, and bind DN password).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

To Synchronize Deletes on Sun DSEE Endpoints

If the destination endpoint in a one-way or bi-directional Sync configuration is a Oracle/Sun DSEE (or Sun DS) server, the Sun DSEE server does not store the value of the user deleting the entry, specified in the modifiersname attribute. It only stores the value of the user who last modified the entry while it still existed. To set up a Sun DSEE destination endpoint to record the user who deleted the entry, you can use the UnboundID Server SDK to create a plug-in as follows:

1. Update the Sun DSEE schema to include a deleted-by-sync auxiliary objectclass. It will only be used as a marker objectclass, so it will not require or allow additional attributes to be present on an entry.

2. Update the Sun DSEE Retro Change Log Plug-in to include the deleted-by-sync auxiliary objectclass as a value for the `deletedEntryAttrs` attribute.

3. Write an LDAPSyncDestinationPlugin script that in the `preDelete()` method modifies the entry that is being deleted to include the `deleted-by-sync` objectclass.

4. Update the Sync Class filter that is excluding changes by the Sync User to also include `(!(objectclass=deleted-by-sync))`.

Configuring DN Maps

Similar to Attribute Maps, DN Maps define mappings when destination DNs differ from source DNs. These differences must be resolved using DN Maps in order for synchronization to successfully take place. For example, the Sync Source could have a DN in the following format:

```
uid=jdoe,ou=People,dc=example,dc=com
```

While the Sync Destination could have the standard X.500 DN format:

- Wildcards. DN Mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can be used only once. The wild card values can be used in the to-dn-pattern attribute using "{1}" to replace their original index position in the pattern, or "{attr}" to match an attribute value. For example:

  ```
  *,**,dc=com->{1},ou=012,o=example,c=us
  ```

  For example, given the DN, uid=johndoe,ou=People,dc=example,dc=com, we want to map the DN to a target DN, uid=johndoe,ou=012,o=example,c=us.

  - "*" matches one RDN component. Thus, "*" matches "uid=johndoe".
  - "**" matches zero or more RDN components. Thus, "**" matches "ou=People,dc=example".
  - "dc=com" matches "dc=com" in the DN.

  The DN is mapped to the "{1},ou=012,o=example,c=us".

- {1} substitutes the first wildcard element. Thus, {1} substitutes "uid=johndoe", so that the DN is successfully mapped to "uid=johndoe,ou=012,o=example,c=us".Regular Expressions. You can also use regular expressions and attributes from the user entry in the to-dn-pattern attribute. For example, the following expression constructs a value for the uid attribute, which is the RDN, out of the initials (first letter of givenname and sn) and the employee ID (the eid attribute) of a user.

  ```
  uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=exampl
  ```

  For more information, see the Configuration Reference Entry DN Map for more details on using regular expression syntax using the to-dn-pattern attribute.

Note: The Synchronization Server automatically validates any DN mapping prior to applying the configuration.

Configuring a DN Map Using dsconfig Interactive

You can use the `dsconfig` tool in interactive mode to create a DN Map. A Sync Class can reference multiple DN Maps. Multiple Sync Classes can share the same DN Map.

To Configure a DN Map Using dsconfig Interactive

1. On the Synchronization Server Configuration console main menu, type the number corresponding to displaying the DN Map Management menu.

2. On the DN Map management menu, type the number corresponding to creating a new DN map.

3. Enter a unique name for the DN Map.

4. For the `from-dn-pattern` property, enter a value. For example, type `**,dc=myexample,dc=com`.

5. For the `to-dn-pattern` property, enter a value. For example, type `{1},o=example.com`.

6. On the DN Map Properties menu, type the number corresponding to entering a general description for the DN Map. This step is optional. Follow the prompts to enter a description for the DN Map. When completed, type `f` to save the changes and apply.

7. After you have configured your DN Mappings, remember to add the new DN Map to a new Sync Class or modify an existing Sync Class.

Configuring a DN Map Using dsconfig Non-Interactive

You can configure a DN Map using the `dsconfig` tool in non-interactive mode that can be included in a setup script in another Synchronization Server installation. Make sure that you understand the mapping process. If you need any assistance, contact your authorized support provider.

To Configure a DN Map Using dsconfig Non-Interactive

1. Use `dsconfig` to create a DN Map for the Synchronization Server.

```
$ bin/dsconfig --no-prompt create-dn-map \
  --map-name nested-to-flattened \
  --set "from-dn-pattern:*,*,dc=example,dc=com" \
  --set "to-dn-pattern:uid={givenname:/^(.)(.*)/\$1/s}{sn:/^(.)(.*)/\$1/s}(eid},
{2},o=example" \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. After you have configured your DN Mappings, remember to add the new DN Map to a new Sync Class or modify an existing Sync Class.

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name test-sync-pipe \
  --class-name test-sync-class \
  --set dn-map:test-dn-map \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configuring Fractional Replication

The UnboundID Synchronization Server supports fractional replication to any type of server. For example, if a replica only performs user authentications, then the Synchronization Server can be configured to propagate, for example, only the `uid` and `userpassword` password policy attributes, reducing the database size at the replica and the network traffic needed to keep this server in sync to this server.

The following example presents a fractional replication use case, where the `uid` and `userPassword` attributes of all entries in the Source topology are synchronized to the Destination topology. Because the `uid` and `userPassword` attributes are present, you also need to synchronize the objectclass attribute. The example assumes that you have already configured a Synchronization Server and defined the sync pipe, sync class, and external servers but have not run realtime synchronization or bulk resync.

To Configure Fractional Replication

1. On the Synchronization Server Configuration console main menu, type the number corresponding to Sync Classes.
2. On the Sync Class management menu, type the number corresponding to viewing and editing an existing Sync Class. Assume that only one Sync Class has been defined thus far.
3. Verify that the Sync Pipe and Sync Class exist.
4. On the Sync Class Properties menu, type the number specifying the source LDAP filter (`include-filter` property) that defines which source entries are to be included in the Sync Class.
5. On the Include-Filter Property menu, type the number corresponding to adding a filter value. For this example, type `(objectclass=person)`. You will prompted to enter another filter. Press **Enter** to continue. On the menu, enter 1 to use the value when specifying it.
6. On the Sync Class Properties menu, type the number corresponding to the `auto-mapped-source-attribute` property. When you change the value from `"-all-"` to a specific attribute, then only the specified attribute is automatically mapped from the Source topology to the Destination topology.
7. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to adding the source attributes that will be automatically mapped to the Destination attributes of the same name. When prompted, enter each attribute, and then press **Enter**.

```
Enter another value for the 'auto-mapped-source-attribute' property
 [continue]: uid
Enter another value for the 'auto-mapped-source-attribute' property
 [continue]: userPassword
Enter another value for the 'auto-mapped-source-attribute' property
 [continue]: objectclass
Enter another value for the 'auto-mapped-source-attribute' property
 [continue]:
```

8. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to removing one or more values. In this example, we want to remove the "`-all-`" value, so that only the `objectclass`, `uid`, and `userPassword` attributes are only synchronized.

9. On the Auto-Mapped-Source-Attribute Property menu, press **Enter** to accept the values.

10. On the Sync Class Properties menu, type the number corresponding to excluding some attributes from the synchronization process. Because we are using the `objectclass=person` filter, we must exclude the `cn`, `givenName`, and `sn` attributes. Enter the menu number corresponding to adding one or more attributes, and then add each attribute that you want to exclude on the `excluded-auto-mapped-source-attributes` Property menu. Here, we want to exclude the `cn`, and `sn` attributes, which are required attributes of the Person objectclass. We also exclude the `givenName` attribute, which is an optional attribute of the `inetOrgPerson` objectclass.

```
Enter another value for the 'excluded-auto-mapped-source-attributes' property
 [continue]: givenName
Enter another value for the 'excluded-auto-mapped-source-attributes' property
 [continue]: sn
Enter another value for the 'excluded-auto-mapped-source-attributes' property
 [continue]:
```

11. On the Excluded-Auto-Mapped-Source-Attributes Property menu, confirm your selections, and then press **Enter** to accept the changes.

Note: If you have a situation where you use entryUUID as a correlation attribute, you may encounter some attribute uniqueness errors while using the `resync` tool. Two ways to fix this are: 1) set the `excluded-auto-mapped-source-attributes` property value to entryUUID on the Sync Class configuration menu, or 2) run `resync` with the `--excludeDestinationAttr` entryUUID argument.

12. On the Sync Class Properties menu, review the configuration, and then type `f` to accept the changes.

13. On the server instances in the Destination topology, you must turn off schema checking due to a schema error that occurs when the required attributes in the Person objectclass are not present. The command assumes that you have already set the global configuration property for the server-group to "`all-servers`". You can use `bin/dsconfig` with the `--applyChangeTo server-group` in non-interactive mode to turn off schema checking on all of the servers in the group.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false --applyChangeTo server-group \
  --port 3389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

14. Run `bin/resync` to load the filtered data from the source endpoint to the target endpoint.

```
$ bin/resync --pipe-name "test-sync-pipe" --numPasses 3
```

15. Run `bin/realtime-sync` to start synchronization.

```
$ bin/realtime-sync start --pipe-name "test-sync-pipe" \
```

```
  --port 7389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

You have successfully configured a fractional replication example.

Managing Failover Behavior

The Synchronization Server delivers high availability in production environments using robust failover mechanisms. To illustrate the generalized failover behavior of the Synchronization Server, Figure 13 shows a simplified synchronization topology with a single failover server on the source, destination, and Synchronization Server, respectively. The gray lines represent possible failover connections in the event the server is down. It is assumed that the external servers are prioritized so that src1 has higher priority than src2; dest1 has higher priority than dest2.

The main Synchronization Server and its redundant failover instance communicate with each other over LDAP and bind using "`cn=IntraSync User,cn=Root DNs,cn=config`". The servers run periodic health checks on each other and share information on all changes that have been processed. Whenever the failover server loses connection to the main Synchronization Server, for example, during a ping or an LDAP search request, it assumes that the main server is down and begins processing changes from the last known change. Control reverts back to the main server once it is back online.

Unlike the Synchronization Servers, the external servers and their corresponding failover server(s) do not run periodic health checks. If an external server goes offline (e.g., dest1), the failover server (e.g., dest2) will receive transactions and remained connected to the Synchronization Server until the Sync Pipe is restarted, regardless if the main external server goes back online.

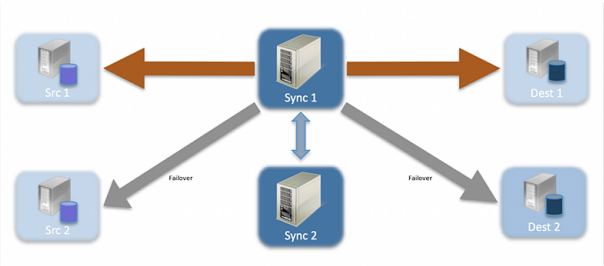


Figure 13: The Synchronization Server in a Simplified Setup

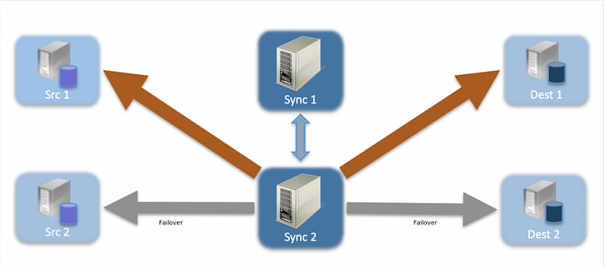


Figure 14: The Synchronization Server during a Failover

Conditions that Trigger Immediate Failover

Immediate failover occurs when the Synchronization Server encounters the following error code that are returned from an external server. The error code numbers are presented in parentheses:

- BUSY (51)
- UNAVAILABLE (52)
- SERVER CONNECTION CLOSED (81)
- CONNECT ERROR (91)

For example, if the Synchronization Server attempts a write operation to a target server (e.g., src1 or dest1) that is in lockdown mode, the Synchronization Server will see a returned UNAVAILABLE error code. The Synchronization Server will then automatically fail over to the next highest prioritized redundant server instance in the target topology (e.g., src2 or dest2), issue an alert, and then reissue the retry attempt. If the operation is unsuccessful for any reason, the server logs the error.

Failover Server Preference

The Synchronization Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the Sync Server prefers to connect to endpoint servers in the same location as itself and also prefers to failover to endpoint servers in the same location as itself. If there are no location settings configured, then the Synchronization Server will simply iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

The Sync Server does not do periodic health checks and will not fail back to a more preferred server automatically. Because of the cost of sync failover (establishing a new connection pool, determining where to pick back up in the changelog, etc.), it will always stay connected to a given server until that server stops responding or until the Sync Pipe is restarted. When a failover does happen, it will always go back to the most preferred server (optionally using location settings to determine this) and work its way down the list. The following provides an example configuration of external servers for illustration purposes.

```
austin1.server.com:1389
london1.server.com:2389
boston1.server.com:3389
austin2.server.com:4389
boston2.server.com:5389
london2.server.com:6389
```

Although they are given descriptive names, these servers do not have their `location` property set and thus will not be able to use location-based failover. If the austin1 server were to become unavailable, the Sync Server will automatically pick up changes on the next server on the list, london1. If london1 is also down, then the next server, boston1 will be picked up. Once the Sync Server iterates through the list, it returns to the top of the list. So, if the Synchronization Server is connected to london2 and it goes down, it will fail over to austin1.

The previous example is not optimal in terms of WAN-friendliness. To minimize WAN traffic, you can configure the location property for each external server using the `dsconfig` command on the Sync Server. We can expand the previous example to include location properties

for each server. Assume that Sync Server has its own `location` property (set in the Global Configuration) set to "austin".

```
austin1.server.com:1389 location=austin
london1.server.com:2389 location=london
boston1.server.com:3389 location=boston
austin2.server.com:4389 location=austin
boston2.server.com:5389 location=boston
london2.server.com:6389 location=london
```

With the `location` property set for each server, the Synchronization Server gets it changes from server austin1. If austin1 goes down, the Sync Server will pick up changes to austin2. If austin2 goes down, then the Sync Server will iterate through the rest of the list in the order it is configured (i.e., london1, boston1, boston2, london2).

The `location` property has another sub-property, `preferred-failover-location` that 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. The `preferred-failover-location` property provides more control over the failover process and allows the failover process to jump to the specified location. Care must be used so that circular failover reference does not take place. In most applications, the `preferred-failover-location` property will not be needed. Here is an example configuration:

```
austin1.server.com:1389 location=austin preferred-failover-location=boston
london1.server.com:2389 location=london preferred-failover-location=austin
boston1.server.com:3389 location=boston preferred-failover-location=london
austin2.server.com:4389 location=austin preferred-failover-location=boston
boston2.server.com:5389 location=boston preferred-failover-location=austin
london2.server.com:6389 location=london preferred-failover-location=london
```

The Sync Server will respect the `preferred-failover-location` if it is set. That is, if it cannot find any external servers in the same location as itself, it will look for any external servers in its own `preferred-failover-location` (in this case, boston). In this example when austin1 becomes unavailable, it will fail over to austin2 because they are in the same location. If austin2 is unavailable, it will fail over to boston1, which is in the `preferred-failover-location` of the Sync Server. If boston1 in unavailable, the Sync Server will fail over to boston2, and finally, it will try the london1 and london2 servers.

Note that any time the Sync Server is currently connected to an endpoint and then loses connectivity, triggering a failover, it will fail over using the preferred server order as determined by location (if set), or else the order that the servers are configured. Using the previous example, this means that it will always try to fail over to austin1 (unless it's failing away from austin1). And then if austin1 cannot be contacted, it will try austin2, boston1, boston2, london1, and finally london2. It will keep cycling through in this order until one can be contacted.

To summarize, external servers with the same location as the Sync Server will be first server to which it fails over, followed by external servers in the preferred failover location of the Sync Server, followed by external servers with no location defined. The sorting is stable; servers within a given location will remain in the same relative order that they started in (in the configured list of external servers). If the Sync Server does not have a location defined, the failover ordering will be determined by the order of the servers that were configured in the list.

Configuration Properties that Control Failover Behavior

The Synchronization Server's out-of-the-box configuration settings should meet the requirements for most applications. Administrators should be aware of four important advanced properties to fine tune the failover mechanism (each property presented in the next section):

- max-operation-attempts (sync pipe)
- response-timeout (source and destination endpoints)
- max-failover-error-code-frequency (source and destination endpoints)
- max-backtrack-replication-latency (source endpoints only)

These properties apply to the following LDAP error codes:

Table 10: LDAP Error Codes

Error Codes	Description
ADMIN_LIMIT_EXCEEDED (11)	Indicates that processing on the requested operation could not continue, because an administrative limit was exceeded.
ALIAS_DEREFERENCING_ PROBLEM (36)	Indicates that a problem was encountered while attempting to dereference an alias for a search operation.
CANCELED (118)	Indicates that a cancel request was successful, or that the specified operation was canceled.
CLIENT_SIDE_LOCAL_ERROR (82)	Indicates that a local (client-side) error occurred.
CLIENT_SIDE_ENCODING_ERROR (83)	Indicates that an error occurred while encoding a request.
CLIENT_SIDE_DECODING_ERROR (84)	Indicates that an error occurred while decoding a request.
CLIENT_SIDE_TIMEOUT (85)	Indicates that a client-side timeout occurred.
CLIENT_SIDE_USER_CANCELLED (88)	Indicates that a user cancelled a client-side operation.
CLIENT_SIDE_NO_MEMORY (90)	Indicates that the client could not obtain enough memory to perform the requested operation.
CLIENT_SIDE_CLIENT_LOOP (96)	Indicates that a referral loop is detected.
CLIENT_SIDE_REFERRAL_LIMIT_ EXCEEDED (97)	Indicates that the referral hop limit was exceeded.
DECODING_ERROR (84)	Indicates that an error occurred while decoding a response.
ENCODING_ERROR (83)	Indicates that an error occurred while encoding a response.
INTERACTIVE_TRANSACTION_ ABORTED (30221001)	Indicates that an interactive transaction was aborted.
LOCAL_ERROR (82)	Indicates that a local error occurred.
LOOP_DETECT (54)	Indicates that a referral or chaining loop was detected while processing a request.
NO_MEMORY (90)	ndicates that not enough memory could be obtained to perform the requested operation.
OPERATIONS_ERROR (1)	Indicates that an internal error prevented the operation from being processed properly.
OTHER (80)	Indicates that an error occurred that does not fall into any of the other categories.

Error Codes	Description
PROTOCOL_ERROR (2)	Indicates that the client sent a malformed or illegal request to the server.
TIME_LIMIT_EXCEEDED (3)	Indicates that a time limit was exceeded while attempting to pro-cess the request.
TIMEOUT (85)	Indicates that a timeout occurred.
UNWILLING_TO_PERFORM (53)	Indicates that the server is unwilling to perform the requested operation.

max-operation-attempts

The `max-operation-attempts` property (part of the Sync Pipe configuration) specifies the maximum number of times to retry a synchronization operation that fails for reasons other than the Sync Destination being busy, unavailable, server connection closed, or connect error.

To Change the max-operation-attempts Property

- To change the default number of retries, use `dsconfig` in non-interactive mode to change the max-operation-attempts value on the Sync Pipe object. The following command changes the number of maximum attempts from 5 (default) to 4. Remember to include the LDAP or LDAPS connection parameters (hostname, port, bindDN, bindDNPassword).

```
$ bin/dsconfig set-sync-pipe-prop --pipe-name "Test Sync Pipe" \
  --set max-operation-attempts:4
```

response-timeout

The `response-timeout` property (part of the Sync Source and Sync Destination configuration) specifies how long the Synchronization Server should wait for a response from a search request to a source server before failing with LDAP result code 85 (client-side timeout). When a client-side timout occurs, the Sync Source will retry the request according to the `max-failover-error-code-frequency` property before failing over to a different source server and performing the retry. The total number of retries will not exceed the max-operation-attempts property defined in the Sync Pipe configuration. A value of zero indicates that there should be no client-side timeout. The default value is one minute.

To Change the response-timeout Property

- To set the `response-timeout` property, use the `dsconfig` tool to set it. Assuming a bidirectional topology, you can set the property on the Sync Source and Sync Destination, respectively. Remember to include the LDAP or LDAPS connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-sync-source-prop --source-name src --set "response-timeout:8 s"

$ bin/dsconfig set-sync-destination-prop --destination-name U4389 --set "response-
timeout:9 s"
```

max-failover-error-code-frequency

The `max-failover-error-code-frequency` property (part of the Sync Source configuration) specifies the maximum time period that an error code can re-appear until it fails over to another server instance. This property allows the retry logic to be tuned, so that retries can be performed once on the same server before giving up and trying another server. The value can be set to zero if there is no acceptable error code frequency and failover should happen immediately. It can also be set to a very small value (such as 10 ms) if a high frequency of error codes is tolerable. The default value is 3 minutes.

To Change the max-failover-error-code-frequency Property

- To change the maximum failover error code frequency, use `dsconfig` in non-interactive mode to change the property on the Sync Source object. The following command changes the frequency from 3 minutes to 2 minutes. Remember to include the LDAP or LDAPS connection parameters (hostname, port, bindDN, bindPassword) with the `dsconfig` command.

```
$ bin/dsconfig set-sync-source-prop --source-name source1 \
  --set "max-failover-error-code-frequency:2 m"
```

max-backtrack-replication-latency

The `max-backtrack-replication-latency` property (part of the Sync Source configuration) sets the time period that a new Synchronization Server will look for any missed changes in the change log to account for any changes that come in due to replication delays. The property should be set to a conservative upper-bound of the maximum replication delay between two servers in the topology. A value of zero implies that there is no limit on the replication latency. The default value is 2 hours. The Synchronization Server stops looking in the change log once it finds a change that is older than the maximum replication latency than the last change it processed on the failed server.

For example, after failing over to another server, the Synchronization Server must look through the new server's change log to find the equivalent place to begin synchronizing any changes. Normally, the Synchronization Server can successfully backtrack with only a few queries of the directory, but in some situations, it might have to look further back through the change log to make sure that no changes were missed. Because the changes can come from a variety of sources (replication, synchronization, and over LDAP), the replicated changes between directory servers are interleaved in each change log. When the Synchronization Server fails over between servers, it has to backtrack to figure out where synchronization can safely pick up the latest changes.

Backtracking occurs until the following:

- It determines that there is no previous change log state available for any source servers, so it must start at the beginning of the change log.

- It finds the last processed replication change sequence number (CSN) from the last time it was connected to that replica, if at all. This process is similar to the "set-startpoint" functionality on the `realtime-sync` tool.
- It finds the last processed replication CSN from every replica that has produced a change so far, and it determines that each change entry in the next oldest batch of changes has already been processed.
- It finds a change that is separated by more than a certain duration (specified by the `max-backtrack-replication-latency` property) from the most recently processed change.

To Change the max-backtrack-replication-latency Property

- To change the maximum backtrack replication, use `dsconfig` in non-interactive mode to change the `max-backtrack-replication-latency` value to some time period. The following command changes the maximum backtracking from two hours to three hours. Remember to include the LDAP or LDAPS connection parameters (hostname, port, bindDN, bindPassword) with this command.

```
$ bin/dsconfig set-sync-source-prop --source-name source1 \
  --set "max-backtrack-replication-latency:3 h"
```

Managing Synchronization Server Extensions

You can create extensions that use the Server SDK to extend the functionality of your Synchronization Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format.

To Run the Manage-Extension Tool

- Run the `manage-extension` tool to install and copy the files. For example, you can install the SCIM extension, `scim-extension-1.1.0`, as follows:

```
$ bin/manage-extension --install scim-extension-1.1.0
```

Chapter 4 Syncing with Active Directory Systems

The UnboundID Synchronization Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory systems. Synchronization with Active Directory systems provides a robust and scalable solution for large multi-directory and multi-national networks. The Synchronization Server also delivers immediate failover capabilities to source and destination instances without data loss in case the target systems go down.

This chapter presents the configuration procedures needed to set up synchronization between UnboundID Directory Server, Alcatel-Lucent 8661 Directory Server, Sun DSEE, or Sun Directory Server source or targets with Microsoft Active Directory systems:

Topics:

- *Overview*
- *Before You Begin*
- *Configuring Active Directory Synchronization*
- *Installing the UnboundID Password Sync Agent*

Overview

The UnboundID Synchronization Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory systems. Synchronization with Active Directory systems provides a robust and scalable solution for large multi-directory and multi-national networks. The Synchronization Server also delivers immediate failover capabilities to source and destination instances without data loss in case the target systems go down.

This chapter presents the configuration procedures needed to set up synchronization between UnboundID Directory Server, Alcatel-Lucent 8661 Directory Server, Sun DSEE, or Sun Directory Server source or targets with Microsoft Active Directory systems:

Before You Begin

Synchronization with Microsoft Active Directory systems requires that SSL be enabled on the Active Directory domain controller, so that the UnboundID Synchronization Server can securely propagate the cn=Sync User account password and other user passwords to the Active Directory target. Likewise, the UnboundID Synchronization must be configured to accept SSL connections.

- For information on setting up an SSL connection on the Synchronization Server, see *Installing the Synchronization Server* on page 25.

Configuring Active Directory Synchronization

To install and configure synchronization with Active Directory systems, you must do the following steps depending on the type of synchronization data:

- On the Synchronization Server, use the `create-sync-pipe-config` tool to configure the Sync Pipes to communicate with the Active Directory source or target.
- If you plan to provide outbound password synchronization from the UnboundID® Directory Server, enable the Password Encryption component on all UnboundID Directory Server sources that receive password modifications. The UnboundID® Directory Server uses the Password Encryption component, analogous to the Password Sync Agent component, to intercept password modifications and add an encrypted attribute, `ds-changelog-encrypted-password`, to the change log entry. The component allows passwords to be synced securely to the Active Directory system, which uses a different password storage scheme. The encrypted attribute appears in the change log and gets synchronized to the other servers but does not appear in the entries. If you do not plan to synchronize passwords, you can skip this step. For more information, see *Configuring the Password Encryption Component*.
- If you plan to provide outbound password synchronization from the Active Directory system, install the Password Sync Agent (PSA) presented in *Installing the UnboundID Password*

Sync Agent after configuring the Synchronization Server. If you do not plan to synchronize passwords, you can skip this step.

To Configure Active Directory Synchronization

The following procedure configures a one-way sync pipe with the Active Directory topology as the Sync Source and the UnboundID Directory Server topology as the Sync Destination.

1. From the `server-root` directory, start the Synchronization Server.

```
$ <server-root>/bin/start-sync-server
```

2. Run the `create-sync-pipe-config` tool to set up the initial Synchronization topology.

```
$ bin/create-sync-pipe-config
```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** to continue the configuration.

4. On the Synchronization Mode menu, press **Enter** to select Standard mode. A standard Mode Sync Pipe will fetch the full entries from both the source and destination and compare them to produce the minimal set of changes to bring the destination into sync. A notification mode Sync Pipe will skip the fetch and compare phases of processing and simply notify the destination that a change has happened and provide it with the details of the change. Notifications are currently only supported from UnboundID and Alcatel-Lucent Directory or Proxy Servers 3.x or later.

5. On the Synchronization Directory menu, select if the Synchronization topology will be one-way (1) or bidirectional (2). In this example, enter "2" for bidirectional.

6. On the Source Endpoint Type menu, enter 6 for Microsoft Active Directory.

```
>>>> Source Endpoint Type

Enter the type of data store for the source endpoint:

    1)  UnboundID Directory Server
    2)  UnboundID Proxy Server
    3)  Alcatel-Lucent Directory Server
    4)  Alcatel-Lucent Proxy Server
    5)  Sun Directory Server
    6)  Microsoft Active Directory
    7)  Microsoft SQL Server
    8)  Oracle Database
    9)  Generic JDBC

    b)  back
    q)  quit

Enter choice [2]:
```

7. On the Source Endpoint Name menu, type a name for the Source Server, or accept the default ("Microsoft Active Directory Source"). For this example, use the "[Microsoft Active Directory]".

8. On the Server Security menu, select the security connection type for the source server, which will be SSL by default for Active Directory configurations. Note that any connection

with the Active Directory topology requires an SSL connection, while connections with the UnboundID Directory Server, Sun DSEE, or Sun Directory Server can use a standard LDAP or SSL connection.

9. On the Servers menu, enter the host name and listener port number for the Source Server, or accept the default (port 636). The server will attempt a connection to the server. If the server is unresponsive, you will be asked to retry <hostname>:636, contact, discard, or keep the server. After entering the first server, enter the additional servers (hostname:port) for the source endpoints, which will be prioritized below the first server. You also have the option to remove any existing servers.

10. On the Synchronization User Account DN menu, enter the User Account DN for the source servers. The account will be used exclusively by the Synchronization Server to communicate with the source external servers. This step will ask you to enter a User Account DN and password, or accept the default account DN (`cn=Sync User,cn=Users,DC=adsync,DC=UnboundID,DC=com`). The User Account DN password must meet the minimum password requirements for Active Directory domains.

11. At this point, you must set up the Destination Endpoint servers. The setup steps are similar to steps 6–11. Select the option for UnboundID and then set up an external destination server and User Account DN.

To Prepare the External Servers

1. After you have configured the Source and Destination Endpoints, the Synchronization Server will prompt you to "prepare" each external server. This step entails asking you if you trust the certificate presented to it, and then testing the connection. The following example shows the user interaction involved in preparing one external server. If you do not prepare the external servers, you can do so after configuring the Sync Pipes using the `prepare-endpoint-server` tool. The following example shows a snippet of a user session:

```
>>>> Prepare Server '10.8.1.163:636'

Servers in a synchronization topology must be 'prepared' for synchronization. This
involves making sure the synchronization user account exists and has the proper
privileges.

Would you like to prepare server '10.8.1.163:636' for synchronization?

1) Yes
2) No

b) back
q) quit

Enter choice [1]:

Testing connection to 10.8.1.163:636

Do you wish to trust the following certificate?
Certificate Subject: CN=WIN-G2R2NXV87VX.adsync.UnboundID.com
Issuer Subject: CN=adsync-WIN-G2R2NXV87VX-CA,DC=adsync,DC=UnboundID,DC=com
Validity: Thus Nov 12 11:39:52 CST 2009 to Fri Nov 12 11:39:52 CST 2010

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

Testing connection to 10.8.1.163:636 ..... Done
Testing 'cn=Sync User,cn=Users,DC=adsync,DC=UnboundID,DC=com' access ..... Done

Configuring this server for synchronization requires manager access. Enter the DN of
```

```
an account capable of managing the external directory server [cn=Administrator,
cn=Users,DC=adsync,DC=UnboundID,DC=com]:

Enter the password for 'cn=Administrator,cn=Users,DC=adsync,DC=unbound,DC=com':
Verifying base DN 'dc=adsync,dc=UnboundID,dc=com' ..... Done
```

2. Next, you will be prompted if you want to prepare another server in the topology. Repeat the process for each server that you have configured in the system.

To Configure the Sync Pipes and its Sync Classes

1. Next, on the Sync Pipe Name menu, you will be prompted to set up the Sync Pipe name. Type a unique name to identify the Sync Pipe or accept the default.

2. On the Pre-Configured Sync Class Configuration for Active Directory Sync Source menu, enter **yes** if you want to synchronize user CREATE operations, and then enter the object class for the user entries at the destination server (default is `inetOrgPerson`). Next, you will be prompted if you want to synchronize user MODIFY and DELETE operations from Active Directory. Enter **yes** if you want to do so.

3. Next, you will be asked if you want to synchronize user passwords from Active Directory. Press **Enter** to accept the default (yes). If you plan to synchronize passwords from Active Directory, you must also install the UnboundID Password Sync Agent component on each domain controller. See *Installing the UnboundID Password Sync Agent* for more information.

4. Next, you will be asked if you want to create a DN map for the user entries in the Sync Pipe. Enter the base DN for the user entries at the Microsoft Active Directory Sync Source (for example, `CN=Users,DC=adsync,DC=UnboundID,DC=com`), and then enter the base DN for the user entries at the UnboundID Directory Server Sync Destination (for example, `OU=users,DC=adsync,DC=UnboundID,DC=com`).

5. At this stage, you will see a list of basic attribute mappings from the Microsoft Active Directory Source to the UnboundID Directory Server destination. If you want to add more complex attribute mappings involving constructed or DN attribute mappings, you must quit the command and use the `dsconfig` tool. The following example shows a sample user session.

```
Below is a list of the basic mappings that have been set up for user entries
synchronized from Microsoft Active Directory -> UnboundID Directory Server. You can
add to or modify this list with any direct attribute mappings. To set up more
complex mappings (such as constructed or DN attribute mappings), use the 'dsconfig'
tool.

   1) cn -> cn
   2) sn -> sn
   3) givenName -> givenName
   4) description -> description
   5) sAMAccountName -> uid
   6) unicodePwd -> userPassword

   b) back
   q) quit
   n) Add a new attribute mapping
```

6. Enter **n** to add a new attribute mapping. First, enter the source attribute, and then enter the destination attribute. The following example shows a sample user session, where a mapping

is set up for the `telephoneNumber` attribute (Active Directory) to the `otherTelephone` attribute (UnboundID).

```
Select an attribute mapping to remove, or choose 'n' to add a new one
[Press ENTER to continue]: n

Enter the name of the source attribute: telephoneNumber

Enter the name of the destination attribute: otherTelephone
```

7. Next, enter **yes** if you want to synchronize group CREATE, MODIFY, and DELETE operations from Active Directory.

8. Next, type **yes** if you want to synchronize group entries from Active Directory. Then, review the basic user group mappings. If you want to use more complex mappings, such as constructed or DN attribute mappings, use the `dsconfig` tool. If you want to add new group attribute mappings, type **n**. In this example, press **Enter** to continue as no new group mappings will be created. The following example shows a portion of a user session.

```
Would you like to sync group entries from Active Directory? (yes / no) [no]: yes
Below is a list of the basic mappings that have been set up for user entries in the
Sync Class: 'AD Groups Sync Class'. You can add to or modify this list with any
direct attribute mappings. To set up more complex mappings (such as constructed or
DN attribute mappings), use the 'dsconfig' tool.

   1) {cn} -> {cn} (Direct Mapping)
   2) 'groupOfUniqueNames' -> {objectClass} (Constructed Mapping)
   3) {member} -> {uniqueMember} (Direct Mapping)

   b) back
   q) quit
   n) Add a new direct attribute mapping

Select an attribute mapping to remove, or choose 'n' to add a new one [Press ENTER
to continue]:
```

9. On the Sync Pipe Sync Class Definitions menu, enter another name for a new Sync class if required. You will essentially repeat the steps in 2–7 to define this new Sync Class. You also have the option to remove any existing sync classes already defined. If you do not require any additional sync class definitions, press **Enter** to continue.

10. Review the Sync Pipe Configuration Summary, and then, press **Enter** to accept the default ("write configuration"), which records the commands in a batch file (`sync-pipe-cfg.txt`). The batch file can be re-used to set up other Sync topologies. The following summary shows two Sync Pipes (from Active Directory to UnboundID; the other, from UnboundID to Active Directory) and its associated Sync Classes.

```
>>>> Configuration Summary
  Sync Pipe: AD to UnboundID
    Source: Microsoft Active Directory
       Type: Microsoft Active Directory
       Access Account: cn=Sync User,cn=Users,DC=adsync,DC=UnboundID,DC=com
       Base DN: DC=adsync,DC=UnboundID,DC=com
       Servers: 10.5.1.149:636

    Destination: UnboundID Directory Server
       Type: UnboundID Directory Server
       Access Account: cn=Sync User,cn=Root DNs,cn=config
       Base DN: dc=example,dc=com
       Servers: localhost:389

    Sync Classes:
       Microsoft Active Directory Users Sync Class
       Base DN: DC=adsync,DC=UnboundID,DC=com
       Filters: (objectClass=user)
       DN Map: **,CN=Users,DC=adsync,DC=UnboundID,DC=com ->{1},ou=users,
```

```
        dc=example,dc=com
       Synchronized Attributes: Custom set of mappings are defined
       Operations: Creates,Deletes,Modifies

  Sync Pipe: UnboundID to AD
    Source: UnboundID Directory Server
       Type: UnboundID Directory Server
       Access Account: cn=Sync User,cn=Root DNs,cn=config
       Base DN: dc=example,dc=com
       Servers: localhost:389

    Destination: Microsoft Active Directory
       Type: Microsoft Active Directory
       Access Account: cn=Sync User,cn=Users,DC=adsync,DC=UnboundID,DC=com
       Base DN: DC=adsync,DC=UnboundID,DC=com
       Servers: 10.5.1.149:636

    Sync Classes:
       UnboundID Directory Server Users Sync Class
       Base DN: dc=example,dc=com
       Filters: (objectClass=inetOrgPerson)
       DN Map: **,ou=users,dc=example,dc=com ->{1},CN=Users,DC=adsync,
        DC=UnboundID,DC=com
       Synchronized Attributes: Custom set of mappings are defined
       Operations: Creates,Deletes,Modifies

    w) write configuration
    b) back
    q) quit

Enter choice [w]:
```

11. Next, you will be prompted as to whether you want to apply the configuration to the local Synchronization Server instance. Type **yes** to apply the configuration changes.

12. Connect to the Synchronization Server by entering the LDAP Connection Parameters: connection type (LDAP, SSL, or StartTLS), host name, port, user bind DN and bind DN password. The configuration is also recorded at `<server-root>/logs/tools/createsync-pipe-config.log`.

To Configure the Password Encryption Component

The next two steps are only required if you are synchronizing passwords from UnboundID Directory Server to Active Directory. They are not required if you are synchronizing from Active Directory to UnboundID Directory Server, or have no plans to synchronize passwords.

1. On the UnboundID Directory Server that will receive the password modifications, enable the `Change Log Password Encryption` component on the directory server. The component intercepts password modifications, encrypts the password and adds an encrypted attribute, ds-changelog-encrypted-password, to the change log entry. You can copy and paste the encryption key from the output if displayed, or you can access it from the <server-root>/bin/sync-pipe-cfg.txt.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Changelog Password Encryption" \
  --set enabled:true --set changelog-password-encryption-key:ej5u9e39pqo68"
```

2. This step is required only if you are syncing passwords from UnboundID Directory Server to Active Directory. It is not required if you are syncing from Active Directory to UnboundID Directory Server, or have no plans to sync passwords. On the Synchronization Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

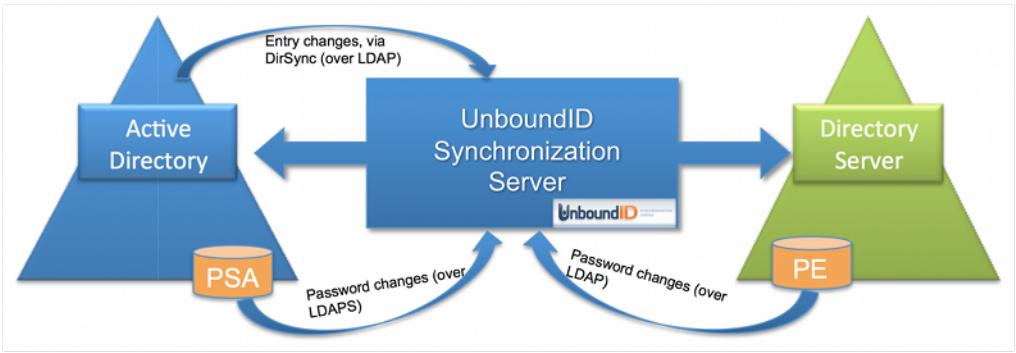
```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pqo68
```

3. You can test the configuration or populate data in the Destination Servers using bulk resync mode. See *Using Resync on the Synchronization Server*. Then, you can use `realtime-sync` to start synchronizing the data. See *Controlling Real Time Synchronization* for more information. Finally, if you are planning on synchronizing passwords, you must install the Password Sync Agent (PSA) on all of the domain controllers in the topology. See the next chapter on how to install the PSA.

Installing the UnboundID Password Sync Agent

When synchronizing passwords with Active Directory systems, the UnboundID Synchronization Server requires that an additional software component, the UnboundID Password Sync Agent (PSA), be installed on all domain controllers in the synchronization topology. This component provides real-time outbound password synchronization from Microsoft Active Directory to any of the Sync Destinations supported by the UnboundID Synchronization Server. Currently these systems include the UnboundID Directory Server, UnboundID Directory Proxy Server (3.x), Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server (3.x), Sun Directory Server 5.x, Sun DSEE (6.x, 7.x), Oracle (10g, 11g), and Microsoft SQL Server (2005, 2008).

The PSA component was designed to provide real-time password synchronization between directories that support differing password storage schemes. The PSA component immediately hashes the password with a 160-bit salted secure hash algorithm and erases the memory where the clear-text password was stored. The component only transmits data over a secure (SSL) connection. The PSA follows Microsoft's security guidelines when handling clear-text passwords (see http://msdn.microsoft.com/en-us/library/ms721884(VS.85).aspx).



Note: For outbound password synchronization from UnboundID Directory Server to Active Directory, you can enable the Password Encryption component, which has similar functionality to that of the PSA component. See *Configuring the Password Encryption Component* for more information.

Figure 15: Password Synchronization with Microsoft Active Directory Systems

The PSA also utilizes Microsoft Windows password filters, which are part of the local security authority (LSA) process. The password filters allow you to implement password policy validation and change notification mechanisms for your system. For more information on this topic, see http://msdn.microsoft.com/en-us/library/ms721882(VS.85).aspx.

The PSA supports failover between Synchronization Servers. It caches the hashed password changes in a local database until it can be guaranteed that all Synchronization Servers in the topology have received them. The failover features provide added flexibility as any or all of the Synchronization Servers can be taken offline or re-configured in real-time without losing any password changes from Active Directory.

The UnboundID Password Sync Agent is safe to leave running on a domain controller indefinitely. If you want to temporarily (or permanently) stop synchronizing passwords, simply remove the `userPassword` attribute mapping in the Synchronization Server, or just stop the Synchronization Server. The PSA will not allow its local cache of password changes to grow out of control; it automatically cleans out records from its local database as soon as they have been acknowledged at a Synchronization Server. It also purges changes that have been in the database for over a week. This feature provides both safety as well as maintenance convenience, so that the PSA will not require any manual updates if the sync configuration changes.

Supported Platforms

The Password Sync Agent (PSA) software has been tested and is supported for the following platforms:

- Windows Server 2003
- Windows Server 2003 R2
- Windows Server 2008
- Windows Server 2008 R2

Before You Install the Password Sync Agent

Before you install the Password Sync Agent, consider the following:

- If you have no plans to synchronize passwords, you do not need to install the PSA.
- Make sure that the Active Directory domain controller has SSL enabled and running on the Windows host machine.
- The UnboundID Synchronization Servers must be configured to accept SSL connections when communicating with the Active Directory host.
- At least one ADSyncSource needs to be configured on the UnboundID Synchronization Server and should point to the domain controller(s) on which the PSA is being installed.
- At the time of installation, all UnboundID Synchronization Servers in the sync topology must be online and available.

- The PSA component is for outbound-only password synchronization from the Active Directory Systems. The PSA component is not necessary if you are performing a one-way password sync from the UnboundID Directory Server to the Active Directory server.

To Install the Password Sync Agent

UnboundID distributes the Password Sync Agent (PSA) in zip file format. The PSA will be available together with each UnboundID Synchronization Server build. Before you install the PSA, ensure that your system meets the conditions presented in the previous section.

The initial (i.e., first time) installation of the PSA requires a system restart for the new PSA DLL (Microsoft requirement).

1. On the domain controller, run the installer by double-clicking the `setup.exe` program.

2. Select an installation folder for the service files. This is where the PSA will store its binaries, local database, and log files.

3. Enter the host names (or IP addresses) and SSL ports of the UnboundID Synchronization Servers. They should be separated by a colon, for example, "sync.host.com:636". Do not add any prefixes to the hostnames.

4. Enter the Directory Manager DN and password. This is only used to set up a special "ADSync" user on the Synchronization Servers. You must also enter a password for this user. Since this is a first-time installation, the ADSync User password will be set to the password you supply. However, if it has been set before (by a previous installation), the password you supply will be verified against the existing password. In this case, make sure you use the same password that was used previously.

5. Enter the maximum size that you would like each log file to grow to in kilobytes. The PSA uses a simple two-file log system. When one reaches the maximum size, it is copied to the backup log file location and a new one is started.

6. Click **Next** to begin the installation. All of the specified Synchronization Servers will be contacted, and any failures will roll back the installation. If everything succeeds, you will see an information message indicating that a restart is required. At this point the PSA is installed but not running. It will not begin until the computer restarts, and the LSA process loads it into memory. Unfortunately, the LSA process cannot be restarted at runtime.

7. If you are syncing pre-encoded passwords from an Active Directory system to an UnboundID Directory system, you must allow pre-encoded passwords in the default password policy.

```
$ bin/dsconfig set-password-policy-prop --policy-name "Default Password Policy" \
--set allow-pre-encoded-passwords:true
```

To Upgrade the Password Sync Agent (restart optional)

For software upgrades for the Password Sync Agent (PSA), the Synchronization Server provides the update tool that upgrades the server code to the latest version. New PSA builds are packaged with the Synchronization Server upgrade distributions. The upgrade does not require Installing

the UnboundID Password Sync Agent 121 a restart, because the core password filter is already running under LSA. The upgrade merely replaces the implementation binaries, which are encapsulated from the password filter DLL.

To upgrade the Password Sync Agent, see Updating the Synchronization Server.

To Uninstall the Password Sync Agent

If you are required to uninstall the PSA, you can simply use the **Add/Remove Programs** on the Windows Control Panel.

1. Go to Control Panel, select **Add/Remove Programs**. Find the Password Sync Agent and click **Remove**. Click **Yes** on the warning that asks if you are sure.

2. At this stage, the PSA has been stopped, and passwords will no longer get synchronized to the synchronization servers or stored in the local database. The implementation DLL has been unloaded, and the database and log files are deleted. Only the binaries remain.

3. The core password filter, however, is still running under the LSA process (it cannot be unloaded at runtime). At this point, it imposes zero overhead on the domain controller, because its implementation DLL has been unloaded. If it is absolutely necessary to remove the password filter itself (located at `C:\WINDOWS\System32\ubidPassFilt.dll`), simply restart the computer. On restart, the password filter and implementation binaries (found in the install folder) can be deleted.

After uninstalling the Password Sync Agent, it cannot be reinstalled without another reboot (i.e. it will revert back to a first-time installation state).

Manual Configuration for Advanced Users

All the configuration settings for the Password Sync Service are stored in the Windows registry under the key `HKLM\SOFTWARE\UnboundID\PasswordSync`. If you want to manually change any of the configuration properties, you can do so at runtime by modifying the values under this registry key. The agent will automatically reload and refresh its settings from the registry. You can verify that the agent is working by checking the current log file, found in the installation directory under `logs\password-sync-current.log`.

Chapter 5 Syncing with Relational Databases

The UnboundID Synchronization Server supports high-scale, highly-available data synchronization between the directory servers and a relational database management systems (RDBMS). UnboundID officially supports synchronization with Oracle Database 10g and 11g as well as Microsoft SQL Server 2005 and 2008. The architecture, however, does not make any assumptions about the type of database or schema being managed; any database with a JDBC 3.0 or higher driver compatible with Java 1.6 can be used.

This chapter presents the following information:

Topics:

- *Overview*
- *About the Server SDK*
- *About the DBSync Process*
- *About the DBSync Example*
- *About the Overall DBSync Configuration Process*
- *Downloading the Software Packages*
- *Creating the JDBC Extension*
- *Configuring the Database for Synchronization*
- *Pre-Configuration Checklist*
- *General Tips When Syncing to a Database Destination*
- *Configuring the Directory-to-Database Sync Pipe*
- *General Tips When Syncing from a Database Source*
- *Configuring the Database-to-Directory Sync Pipe*
- *Synchronizing a Specific List of Database Elements Using Resync*

Overview

The UnboundID Synchronization Server supports high-scale, highly-available data synchronization between the directory servers and a relational database management systems (RDBMS). UnboundID officially supports synchronization with Oracle Database 10g and 11g as well as Microsoft SQL Server 2005 and 2008. The architecture, however, does not make any assumptions about the type of database or schema being managed; any database with a JDBC 3.0 or higher driver compatible with Java 1.6 can be used.

About the Server SDK

To synchronize LDAP data to or from a relational database, you must first create a JDBC Sync Source or Destination extension to act as an interface between the UnboundID Synchronization Server and your database environment. You can create the extension using the UnboundID Server SDK, which provides APIs to develop plug-ins and third-party extensions to the server using Java or Groovy. The Server SDK's documentation (javadoc and examples) is delivered with the Server SDK build in zip format.

Note: Server SDK support is provided if you have purchased Premium Support for the product that you are developing extensions for. However, UnboundID does not provide support for the third party extensions developed using the Server SDK. You should contact your product level support group if you need assistance.

The Server SDK contains two abstract classes that will likely be required for your implementation:

- `com.unboundid.directory.sdk.sync.api.JDBCSyncSource`
- `com.unboundid.directory.sdk.sync.api.JDBCSyncDestination`

The abstract classes correspond to how you use the database, either as a source of synchronization or as a target destination. The remainder of the SDK contains helper classes and utility functions to make the script implementation simpler.

The SDK allows you to use any change tracking mechanism to detect changes in the database. However, we provide a recommended generic approach using a simplified change log table and triggers to record changes. Our solution is largely vendor and database version-independent and is configurable on any database that supports row-based trigger semantics. Examples are provided in the `config/jdbc/samples` directory for Oracle Database and Microsoft SQL Server.

The Synchronization Server uses the scripted adapter layer (shown in Figure 16) to convert any database change to an equivalent LDAP entry. From there, the Sync Pipe processes the data through inclusive (or exclusive) filtering together using attribute and DN maps defined in the Sync Classes to update the endpoint servers. For example, once you have implemented

a script using Java, you can configure it for use by setting the extension-class property on a ThirdPartyJDBCSyncSource or ThirdPartyJDBCSyncDestination configuration object within the Synchronization Server. The procedures to accomplish this are presented later in this chapter.

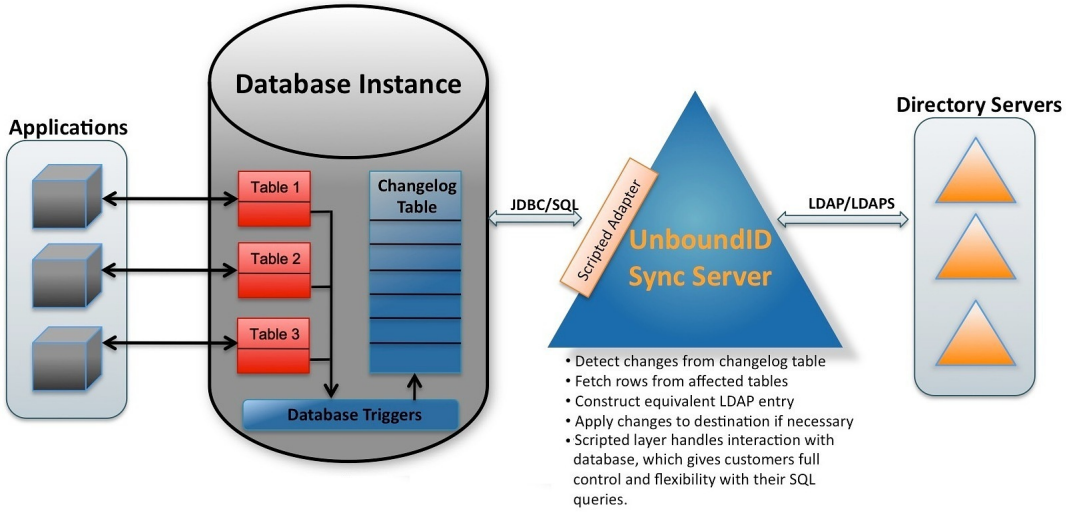


Figure 16: Architectural Overview

About the DBSync Process

The Synchronization Server is designed for high-scale, point-to-point data synchronization between a directory server and a RDBMS system via an UnboundID Server SDK extension. The Synchronization Server provides multiple configuration options, such as advanced filtering (fractional and subtree), attribute and DN mappings, transformations, correlations, and configurable logging features for seamless one-way or bidirectional synchronization.

To support synchronizing changes out of a database, the database must be configured with a change tracking mechanism. We recommend a general approach involving triggers (one trigger per table) to record all changes to a change log table. The database change log table should record the type of change (`INSERT`, `UPDATE`, `DELETE`) that occurred, the specific table name, the unique identifier for the row that was changed, the database entry type, the changed columns, the modifier's name, and the timestamp of the change. An example is presented later in this chapter.

The Synchronization Server delegates the physical interaction with the database to a userdefined extension, which has full control of the SQL queries. The extension layer provides flexibility in how you define the mapping semantics between your LDAP environment and your relational database environment. The connection management, pooling, retry logic, and other boilerplate code are all handled internally by the Synchronization Server.

The RDBMS Synchronization (DBSync) implementation does not support failover between different physical database servers as is the case for directory servers. Most enterprise databases

have a built-in failover layer (for example, Microsoft's node-based SQL Server Failover Cluster) from which the Synchronization Server can point to a single virtual address and port and still be highly available. Note that while you can have a single RDBMS node, you can scale to multiple directory server instances at the other endpoint.

About the DBSync Example

The Synchronization Server provides a DBSync example between two endpoints consisting of an UnboundID Directory Server source and a RDBMS system, which will be used throughout this chapter. The entity-relational diagram for the normalized database schema is available in `/config/jdbc/samples/oracle-db/ComplexSchema.jpg` and is shown in Figure 17. Five tables are represented with their primary keys in bold. The entity relations and foreign keys are marked by the relationship lines.

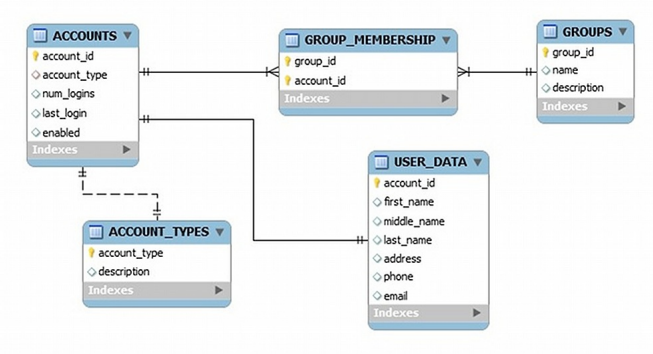


Figure 17: ER Diagrams for the Schema Tables

Example DS Entries

The synchronization example assumes that the directory server's schema has been configured to handle the mapped attributes. If you are configuring a database-to-directory sync pipe with a newly installed directory server, you must ensure that the schema has the correct `attributeType` and `objectClass` definitions in place. You can add the definitions in a custom 99-user.ldif file in the `config/schema` folder of your directory server implementation, if necessary. The following snippet shows an example of the LDAP entries that are used in the synchronization example. Figure 17 maps to a custom object class on the directory server, while the "groups" table maps to a standard LDAP group entry with `objectclass:groupOfUniqueNames`. Example entries appear as follows:

```
dn: accountid=0,ou=People,dc=example,dc=com
objectClass: site-user
firstName: John
lastName: Smith
accountID: 1234
email: jsmith@example.com
phone: +1 556 805 4454
address: 17121 Green Street
numLogins: 4
lastLogin: 20070408182813.196Z
enabled: true

dn: cn=Group 1,ou=Groups,dc=example,dc=com
objectClass: groupOfUniqueNames
description: This is Group 1
```

```
uniqueMember: accountID=0,ou=People,dc=example,dc=com
uniqueMember: accountID=1,ou=People,dc=example,dc=com
```

About the Overall DBSync Configuration Process

The procedure to configure a DBSync system is slightly more complicated than that of a directory server-to-directory server synchronization configuration due to the extra tasks required to create the extensions and to configure the database. The overall configuration process is as follows:

1. Download the appropriate JDBC 3.0 or higher driver. UnboundID cannot bundle any JDBC drivers with the Synchronization Server due to licensing restrictions, but most are freely available to end users. When ready, you can place it in the lib directory in the Synchronization Server's installation directory (`/UnboundID-Sync/lib`), and then restart the server for the driver to get loaded into the runtime.

   For example, you should download the `ojdbc6.jar` file for Oracle systems or the `sqljdbc4.jar` file for MS SQL Server systems.

   Older JDBC drivers which do not implement the Java Service Provider mechanism may have to specify "jdbc.drivers" as a JVM argument. Open the java.properties file using a text editor, add the `jdbc.drivers` argument to a utility, save the file, and then run the `dsjavaproperties` command to apply the change. For example, enter the following for `start-syncserver`:

   ```
   start-sync-server.java-args=-d64 -server -Xmx256m -Xms256m
   -XX:+UseConcMarkSweepGC -
   Djdbc.drivers=foo.bah.Driver:wombat.sql.Driver:com.example.OurDriver
   ... etc.
   ```

   See the documentation for DriverManager (`http://download.oracle.com/javase/6/docs/api/java/sql/DriverManager.html`) for more information.

2. Create one or more JDBC extensions based on the Server SDK. If you are configuring for bidirectional synchronization, you will need two scripts: one for the JDBC Sync Source; the other for the JDBC Sync Destination. Place the compiled extension in the `/lib/extensions`.

3. Configure the database change log table and triggers (presented later). While you can use the vendor's native change tracking mechanism, we recommend setting up a change log table, shown later in the configuration procedures. Each table requires one database trigger to detect the changes and loads them into the change log table.

4. Configure the Sync Pipes including the Sync Classes, external servers, DN and attribute maps for one direction (e.g., from directory server to database).

5. Run the `resync --dry-run` command to test the configuration settings.

6. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.

7. Run the `resync` command to populate data on the destination endpoint.

8. Start the Sync Pipes using the `realtime-sync start` command.

9. Monitor the synchronization server using the `status` commands and logs.

10. For bidirectional synchronization, configure another Sync Pipe in the other direction (e.g., from database to directory server), repeat steps 4–8 to test the complete synchronization system.

Downloading the Software Packages

You need to download your JDBC driver prior to setting up your synchronization environment with a RDBMS system.

1. Download the UnboundID Synchronization Server ZIP file. Unzip the server in a directory of your choice.

2. If you are configuring the Synchronization Server from scratch, you must ensure that you have JDK1.6 update 25. The JDK is required to build any Server SDK extensions. Set the `JAVA_HOME` environment variable and your `PATH` or `CLASSPATH` variables accordingly.

   Download an appropriate JDBC 3.0 or higher driver for your system. Place it in the lib directory in the Synchronization Server's installation directory (`/UnboundID-Sync/lib`). For example, you should download the `ojdbc6.jar` file for Oracle systems or the `sqljdbc4.jar` file for MS SQL Server systems.

   You will need to re-start the server to pick up changes to an extension.

3. Download the Server SDK zip file and unzip it in a directory of your choice.

Creating the JDBC Extension

The JDBC extension implementation must be written in Java or the Groovy scripting language (`http://groovy.codehaus.org/api/`). Consult the Server SDK documentation for details on how to build and deploy extensions. The examples in this guide use pure Java. Both languages have been tested and are fully supported by the Synchronization Server. UnboundID recommends implementing extensions in pure Java, because it is more strict and will catch programming errors during compile time rather than at runtime. Groovy is more flexible and can accomplish more with less lines of code, but it also can be more difficult to understand.

For those unfamiliar with Groovy, it is an open-source, dynamically-typed scripting language, similar to Java and provides quick adoption for those developers who already know the Java programming language. Groovy scripts can leverage existing Java classes and libraries to allow embedded applications within Java or as standalone scripts. Extensions written in Groovy are loaded at Sync Pipe startup, which allows you to dynamically reload a script by restarting the Sync Pipe.

Groovy scripts must live under the `/lib/groovy-scripted-extensions` directory (Java implementations using the Server SDK reside under `lib/extensions`), which may also contain other plug-ins built using the UnboundID Server SDK. If a script declares a package name, it must live under the corresponding folder hierarchy, just like a Java class. For example, to use a script class called `ComplexJDBCSyncSource` whose package is `com.unboundid.examples.oracle`, place it under the `/lib/groovy-scripted-extensions/com/unboundid/examples/oracle` and set the `script-class` property on the Sync Source

to `com.unboundid.examples.oracle.ComplexJDBCSyncSource`. There are a few reference implementations provided in the `config/jdbc/samples` directory. You can use the `manage-extension` tool in the `bin` directory (UNIX/LINUX) or `bat` directory (Windows) to install or update the extension. See the *Managing Extensions* section for more information.

Note: Any changes to an existing script requires a manual Sync Pipe restart. Any configuration change automatically restarts the affected Sync Pipe.

The default libraries available on the classpath to your script implementation include:

- Groovy 1.7 (`http://groovy.codehaus.org/api/`)
- UnboundID LDAP SDK for Java 2.2.0 (`http://unboundid.com/products/ldapsdk/docs/javadoc/index.html`)
- JRE 1.6 (`http://download.oracle.com/javase/6/docs/api/`)

Logging from within a script can be done with the Server SDK's `ServerContext` abstract class. Some of ServerContext's methods, such as `registerChangeListener()` or `getInternalConnection()` will not be available when running the `Resync` tool, because it runs outside of the Synchronization Server process. Any logging performed within a script during a Resync operation will appear in the `logs/tools/resync.log` file.

About Groovy

There are a few things to be aware of when using Groovy:

- Semicolons are optional.
- The 'return' keyword is optional.
- You can use the 'this' keyword inside static methods (which refers to this class).
- Methods and classes are public by default.
- The 'throws' clause in a method signature is not checked by the Groovy compiler, because there is no difference between checked and unchecked exceptions.
- You will not get compile errors like you would in Java for using undefined members or passing arguments of the wrong type.
- Arrays need to be declared with square brackets (for example, `int[] myArray = [1,2,3]`). See the reference at `http://groovy.codehaus.org/Differences+from+Java`.

Implementing a JDBC Sync Source

The JDBCSyncSource abstract class must be implemented to synchronize data out of a relational database (e.g., for database to directory server synchronization). Since the UnboundID Synchronization Server is LDAP-centric, this class allows you to take database content and

convert it into LDAP entries. For more detailed information on the class, consult the UnboundID Server SDK Javadoc.

The extension imports classes from the Java API, UnboundID LDAP SDK for Java API, and the UnboundID Server SDK. Depending on the data, you will need to implement the following methods within your script:

- **initializeJDBCSyncSource**. Called when a Sync Pipe first starts up, or when the Resync process first starts up. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- **finalizeJDBCSyncSource**. Called when a Sync Pipe shuts down, or when the Resync process shuts down. Any clean up should be performed here, and all internal resources should be freed.
- **setStartpoint**. Sets the starting point for synchronization by identifying the starting point in the change log. This method should cause all changes previous to the specified start point to be disregarded and only changes after that point to be returned by the `getNextBatchOfChanges` method. There are several different startpoint types (see SetStartpointOptions in the Server SDK), and this implementation is not required to support them all. If the specified startpoint type is unsupported, this method throws an exception (`IllegalArgumentException`). This method can be called from two different contexts: when the `realtime-sync set-startpoint` command is used (the Sync Pipe is required to be stopped in this context) or immediately after a connection is first established to the source server (e.g., before the first call to `getNextBatchOfChanges` method).

Note: The `RESUME_AT_SERIALIZABLE` startpoint type must be supported by your implementation, because this method is used when a Sync Pipe first starts up and loads its state from disk.

- **getStartpoint**. Gets the current value of the startpoint for change detection.
- **fetchEntry**. Returns a full source entry (in LDAP form) from the database, corresponding to the DatabaseChangeRecord object that is passed in. The resync command also uses this class to retrieve entries.
- **acknowledgeCompletedOps**. Provides a means for the Synchronization Server to acknowledge to the database which operations have completed processing.

Note: The internal value for the startpoint should only be updated after a sync operation is acknowledged back to this script (via this method). Otherwise it will be possible for changes to be missed when the Synchronization Server is restarted or a connection error occurs.

- **getNextBatchOfChanges**. Retrieves the next set of changes for processing. The method also provides a generic means to limit the size of the result set.
- **listAllEntries**. Used by the resync command to get a listing of all entries.

- **cleanupChangelog**. In general, we recommend implementing a cleanupChangelog method, so that the Synchronization Server can purge old records from the change log table, based on a configurable age.

See the `config/jdbc/samples` directory for example script implementations and the Server SDK javadoc for more detailed information on each method.

Implementing a JDBC Sync Destination

The JDBCSyncDestination abstract class must be implemented to synchronize data into a relational database (e.g., for directory server to database synchronization). The class allows you to take LDAP content and convert it to database content.

The extension imports classes from the Java API, UnboundID LDAP SDK for Java API, and the UnboundID Server SDK, depending on your database configuration. You will need to implement the following methods within your script:

- **initializeJDBCSyncDestination**. Called when a Sync Pipe first starts up, or when the Resync process first starts up. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- **finalizeJDBCSyncDestination**. Called when a Sync Pipe shuts down, or when the Resync process shuts down. Any clean up should be performed here, and all internal resources should be freed.
- **createEntry**. Creates a full database entry (or row), corresponding to the LDAP Entry that is passed in.
- **modifyEntry**. Modify a database entry, corresponding to the LDAP Entry that is passed in.
- **fetchEntry**. Return a full destination database entry (in LDAP form), corresponding to the source entry that is passed in.
- **deleteEntry**. Delete a full entry from the database, corresponding to the LDAP Entry that is passed in.

For more detailed information on the abstract class, consult the *Server SDK Javadoc*.

Configuring the Database for Synchronization

To configure the database for synchronization, you must do three things: 1) set up a database SyncUser account; 2) set up the change tracking mechanism; and 3) set up the database triggers (one per table) for your application. The following example uses the example setup script is available in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`, where items in brackets (for example `[ubid_changelog]`) is a user-named label for the account, table or column.

Note: Database change tracking is only necessary if you are syncing FROM the database. If you are syncing TO a database, you only need to set up the SyncUser account and the correct privileges.

1. Create an Oracle login (`SyncUser`) for the Synchronization Server, so that the Synchronization Server can access the database server. Also make sure to grant sufficient privileges to the SyncUser for any tables to be synchronized. Make sure to change the default password on production systems.

```
CREATE USER SyncUser IDENTIFIED BY password
  DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
GRANT "RESOURCE" TO SyncUser;
GRANT "CONNECT" TO SyncUser;
```

2. Set up your change log tables on the database. An example is presented as follows:

```
CREATE TABLE ubid_changelog (
  --This is the unique number for the change change_number Number NOT NULL PRIMARY
 KEY,
  --This is the type of change (insert, update, delete). NOTE: This should represent
  --the actual type of change that needs to happen on the destination(for example a
  --database delete might translate to a LDAPmodify, etc.)
  change_type VARCHAR2(10) NOT NULL,

  --This is the name of the table that was changed table_name VARCHAR(50) NOT NULL,
  --This is the unique identifier for the row that was changed. It is up to
  --the trigger code to construct this, but it should follow a DN-like format
  --(e.g. accountID={accountID}) where at least the primary key(s) are
  --present. If multiple primary keys are required, they should be delimited
  --with a unique string, such as '%%' (e.g. accountID={accountID}%%
  --groupID={groupID})
  identifier VARCHAR2(100) NOT NULL,

  --This is the database entry type. The allowable values for this must be
  --set on the JDBC Sync Source configuration within the Synchronization
  --Server.
  entry_type VARCHAR2(50) NOT NULL,

  --This is a comma-separated list of columns that were updated as part of
  --this change.
  changed_columns VARCHAR2(1000) NULL,

  --This is the name of the database user who made the change
  modifiers_name VARCHAR2(50) NOT NULL,

  --This is the timestamp of the change
  change_time TIMESTAMP(3) NOT NULL, CONSTRAINT chk_change_type
    CHECK (change_type IN ('insert','update','delete'))) ORGANIZATION INDEX;
```

3. Create an Oracle function to get the SyncUser name. This is a convenience function for the triggers.

```
CREATE OR REPLACE FUNCTION get_sync_user RETURN VARCHAR2
IS
BEGIN
  RETURN 'SyncUser';
END get_sync_user;
```

4. Create an Oracle sequence object for the change-number column in the change log table.

```
CREATE SEQUENCE ubid_changelog_seq MINVALUE 1 START WITH 1
NOMAXVALUE INCREMENT BY 1 CACHE 100 NOCYCLE;
```

5. Create a Database Trigger for each table that will participate in synchronization. An example is shown below and shows a trigger for the Accounts table that tracks all changed columns after any `INSERT`, `UPDATE`, and `DELETE` operation. The code generates a list of changed items and then inserts them into the change log table. See the example in `/config/ jdbc/ samples/oracle-db/OracleSyncSetup.sql`.

```
CREATE OR REPLACE TRIGGER ubid_accounts_trg AFTER INSERT OR
  DELETE OR UPDATE ON accounts
FOR EACH ROW
DECLARE
  my_identifier ubid_changelog.identifier%TYPE;
  my_changetype ubid_changelog.change_type%TYPE;
  my_changedcolumns ubid_changelog.changed_columns%TYPE := '';
  CURSOR column_cursor IS select COLUMN_NAME from USER_TAB_COLUMNS where
    TABLE_NAME='ACCOUNTS';
BEGIN
  --Short circuit and do nothing if the change came from the Synchronization Server
 itself.
  --This prevents loopbacks when doing bidirectional synchronization.

  IF UPPER(USER) = UPPER(get_sync_user()) THEN RETURN; END IF;

  -- Figure out change type
  IF INSERTING THEN
    my_identifier := 'accountID=' || :NEW.accountID;
    my_changetype := 'insert';
  ELSIF DELETING THEN
    my_identifier := 'accountID=' || :OLD.accountID;
    my_changetype := 'delete';
  ELSIF UPDATING THEN
    my_identifier := 'accountID=' || :NEW.accountID;
    my_changetype := 'update';

    -- Figure out changed coumns
    FOR my_row IN column_cursor
      LOOP
        IF UPDATING (my_row.COLUMN_NAME) THEN
          my_changedcolumns := my_changedcolumns || my_row.COLUMN_NAME || ',';
        END IF;
      END LOOP;
  END IF;

  --Do the insert
  INSERT INTO ubid_changelog (change_number, change_type, table_name, identifier,
 entry_type,
    changed_columns, modifiers_name, change_time) VALUES
(ubid_changelog_seq.NEXTVAL,
    my_changetype, 'ACCOUNTS', my_identifier, 'account', my_changedcolumns, USER,
 SYSTIMESTAMP);

  --If changes to this table affect multiple LDAP entries, multiple records should
  --be inserted into the changelog table. For example, if an update to an "account"
in
  --the database affected an "account" LDAP entry and a "groups" LDAP entry, then we
  --would have another "INSERT INTO ubid_changelog..." here with a different entry
  --type.
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Changelog trigger exception:');
      DBMS_OUTPUT.PUT_LINE(DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
  END;
```

Pre-Configuration Checklist

Before configuring the Synchronization Server, we assume that you have accomplished the following items:

- Create a Sync User account with the access privileges to the RDBMS server, so that the Synchronization Server can access the machine.

- Set up your change log tables on the database.
- Set up the triggers in the database: one per table that will participate in synchronization.
- Create a JDBC extension using Java or Groovy to map the LDAP Entries to the RDBMS table rows or vice-versa, place it in the `/lib/groovy-scripted-extensions` directory (Java implementations using the Server SDK reside under lib/extensions). You must also create an extension if configuring a Sync Pipe from database to directory server. See the example in the `config/jdbc/samples` directory.
- Make sure the database is configured to listen for external connections. If it is locked down for security, the Synchronization Server will register a connection error during Sync Pipe startup.

General Tips When Syncing to a Database Destination

When configuring a directory-to-database Sync Pipe, you should be aware of the following recommendations:

- **Identify the Object Classes**. It is advisable to identify the different object classes that will be synced. Create a Sync Class per object class, so that you can easily distinguish between them and have different mappings and sync rules set up for each one.
- **For each Sync Class**, make sure to set the following items listed below. You can access many of the configuration menus using the `dsconfig` tool.
  - **Set the Include-Filter Property**. Make sure the include-filter property is set on the Sync Class configuration menu to something that will uniquely identify the source entries, such as "objectClass=customer".
  - **Create Specific Attribute Mappings**. Create a specific Attribute Mapping for every LDAP attribute that you want to be synced to a database column(s); add all of these to a single Attribute Map and set it on the Sync Class. This way, the script will not have to know about the schema on the directory side. It may be desirable to add a Constructed Attribute Mapping that maps a literal value to the `objectClass` attribute, if needed by the script, to determine the database entry type. For example, you could have something like "account" -> objectClass, which would result in the constructed destination LDAP entry always containing an objectClass of "account".
  - **Create Specific DN Maps** (optional). Create a DN Map that recognizes the DN's of the source entries and maps them to a desired destination DN. In most cases, this step is unnecessary, because the script will use the attributes rather than the DN to figure out which database entry needs changed.
  - **Set auto-mapped-source-attribute to "-none-"**. Remove the default value of "-all-" from "auto-mapped-source-attribute" on the Sync Class configuration menu, and replace it with the value "-none-". We do not want any values from the directory automatically mapped to an attribute with the same name when using explicit attribute mappings. (You can set this property using the `dsconfig` tool. On the UnboundID Synchronization Server configuration console main menu, select Sync Class, and then enter View edit an existing

sync class on the Sync Class Management menu. Select your Sync Class to open the Sync Class Configuration menu.)

- **Configure Create-Only Attributes**. Any attributes that should be included on creates but never modified (such as objectclass) should be specified on the Sync Pipe as a `create-only` attribute. This way if the Synchronization Server ever computes a difference in that attribute between the source and destination, it will not try to modify it at the destination.

- **Avoid bidirectional Loopback**. Make sure to set the `ignore-changes-by-[user|dn]` property on both Sync Sources configuration menus when configuring for bidirectional synchronization. This is important to make sure that changes are not looped back by the Synchronization Server.

- **Synchronizing DELETE Operations**. On UnboundID Directory Server and Alcatel-Lucent 8661 Directory Server systems, you must configure the `changelog-deleted-entryinclude-attribute` property on the Change Log backend menu using the `dsconfig` tool. This property allows for the proper synchronization of `DELETE` operations that occur with this endpoint server. For the example presented in this section, you would set the `changelog-deleted-entry-include-attribute=accountid`. For more information, see *Configuring the Directory Server Backend for Synchronizing Deletes*.

- **Set the Attribute-Synchronization-Mode Appropriately for DB Sync.** For `MODIFY` operations, the Synchronization Server detects any change on the source change log, fetches the source entry, applies mappings, computes the equivalent destination entry, fetches the actual destination entry, and then runs a diff between the two entries to determine the minimal set of changes to get the destination in sync with the source. By default, the Synchronization Server only makes changes on the destination entry for those attributes that were detected in the original change log entry. However, this is configurable using the `attribute-synchronization-mode` property. The `attribute-synchronization-mode` property sets the type of diff operation that is performed between the source and destination entries on a `MODIFY` operation, which in turn determines the scope of attributes that are modified on the destination.

  If the source endpoint is a database server (Oracle or MS SQL Server), we recommend setting the `attribute-synchronization-mode` property to `all-attributes` on the Sync Class configuration menu. In this way, the `diff` operation will consider all the source attributes and any that have changed will be updated on the destination, even if the change was not originally detected in the change log. In some cases, you may not be able to get a list of changed columns in the database, in which case, you would have to use this mode, because `modified-attributes-only` will not change any destination attributes if it thinks that there are no source attributes changed. If both endpoints are directory servers, we recommend keeping the default configuration to `modified-attributes-only` to avoid any possible replication conflicts.

- **Handling MODDN Operations**. The concept of a modifyDN or renaming an entry does not have a direct equivalent in the relational database world. The `JDBCSyncDestination` API does not provide a separate method for handling changes of this type; instead, the `modifyEntry()` method is called just as if it is a normal change. The extension can check if the entry was renamed by looking at the `SyncOperation` that is passed in (i.e., `syncOperation.isModifyDN()`). If this method returns true, the `fetchedDestEntry` parameter will have the old DN; the new DN can be obtained by calling `syncOperation.getDestinationEntryAfterChange()` and getting the DN from there.

Configuring the Directory-to-Database Sync Pipe

The following procedure shows the interactive steps to set up a one-way Sync Pipe with an UnboundID Directory Server as the Sync Source and a RDBMS (Oracle) system as the Sync Destination. The procedure uses the create-sync-pipe-config tool in interactive commandline mode, which shows the configuration steps in a top-down flow from Sync Pipe to external servers.

The procedure is broken out into sections for easy access and is based on the interactive prompts that the create-sync-pipe-config tool will present. The instructions assume that the user has the proper root user or admin privileges to make configuration changes. Once you have configured the sync pipes, then you can fine-tune the configuration later using the dsconfig utility.

Step 1. Creating the Directory-to-Database Sync Pipe

The initial configuration steps show how to set up a single Sync Pipe from a directory server instance to a database using the `create-sync-pipe-config` tool in interactive mode. The `create-sync-pipe-config` tool prompts the user for input and leads you through the configuration steps in a wizard-like mode. The procedure will show how to set up and configure the Sync Pipe, external servers, and Sync Classes. The examples are based on the Complex JDBC sample in the `config/jdbc/samples/oracle-db` directory.

Optionally, you can run the `create-sync-pipe-config` tool with the server offline and import the configuration later.

1. Start the Synchronization Server.

```
$ bin/start-sync-server
```

2. Run the `create-sync-pipe-config` tool.

```
$ bin/create-sync-pipe-config
```

3. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.

4. On the Synchronization Mode menu, press **Enter** to select Standard mode. A standard Mode Sync Pipe will fetch the full entries from both the source and destination and compare them to produce the minimal set of changes to bring the destination into sync. A notification mode Sync Pipe will skip the fetch and compare phases of processing and simply notify the destination that a change has happened and provide it with the details of the change. Notifications are currently only supported from UnboundID and Alcatel-Lucent Directory or Proxy Servers 3.0.3 or later.

5. On the Synchronization Directory menu, enter the number corresponding to Create a One-way Sync Pipe from directory to database. If you are planning to deploy a bidirectional Sync configuration, enter the number corresponding to bidirectional synchronization.

To Configure the Sync Source

1. On the Source Endpoint Type menu, enter the number for the sync source corresponding to the type of source external server. For this example, enter the number corresponding to the UnboundID Directory Server.

2. Next, you will be prompted to enter a name for the Source Endpoint. Enter a descriptive name for the Sync Source. For example, `dssync`.

3. Next, enter the base DN for the directory server, which is used as the base for LDAP searches. For example, enter `dc=example,dc=com`, and then press **Enter** again to return to the menu. If you enter more than one base DN, make sure the DNs do not overlap.

4. On the Server Security menu, select the type of secure communication that the Synchronization Server will use with the endpoint server instances. Select either 1) None; 2) SSL; or 3) StartTLS. For this example, select the default (None).

5. Next, enter the host and port of the first Source Endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The Synchronization Server will contact this first server if it is available, then contact the next highest priority server if the first server is unavailable, etc. After you have entered the host and port, the Synchronization Server tests that a connection can be established.

6. On the Synchronization Server User Account, enter the DN of the sync user account and create a password for this account. The Sync User account allows the Synchronization Server to access the source endpoint server. By default, the Sync User account is placed at `cn=Sync User,cn=Root DNs,cn=config`. Press **Enter** to accept the default configuration.

To Configure the Destination Endpoint Server

1. Next, on the Destination Endpoint Type menu, select the type of datastore on the endpoint server. In this example, enter the number corresponding for Oracle Database.

2. Next, you will be prompted to enter a name for the Destination Endpoint. Enter a descriptive name for the Sync Destination. For example, oraclesync.

3. On the JDBC Endpoint Connection Parameters menu, enter the fully-qualified and resolvable host name or IP address for the Oracle database server. After you have entered the host name, the Synchronization Server checks if the hostname or IP address is resolvable.

4. Next, enter the listener port for the database server. For this example, press Enter to accept the default (1521).

5. Enter a database name. For this example, use `dbsync-test`.

6. Next, the Synchronization Server attempts to locate the JDBC driver in the lib directory. If the server found the file, it will generate a success message.

```
Successfully found and loaded JDBC driver for:
jdbc:oracle:thin:@//dbsync-w2k8-vm-2:1521/dbsync-test
```

If the server cannot find the JDBC driver, you can add it later, or quit the `create-sync-pipe-config` tool and add the file to the lib directory. The following message is displayed to std-out.

```
Could not find an appropriate JDBC driver in the /UnboundID-Sync/lib
directory for: jdbc:oracle:thin:@//dbsync-w2k8-vm-2:1521/dbsync-test

What do you want to do?

    1)  I will add the JDBC driver later
    2)  Quit this tool and add the JDBC driver now

    b)  back
    q)  quit
Choose an option [1]:
```

7. Next, you will be prompted if you want to add any additional JDBC connection properties for the database server. Please consult your JDBC driver's vendor documentation to see what properties are supported. For this example, press Enter to accept the default (no).

8. Next, you will be prompted to enter a name for the database user account with which the Synchronization Server will communicate. Press Enter to accept the default (SyncUser). Then, enter the password for the SyncUser account. For information on creating the SyncUser account on the Oracle Server, see step 1 in *Configuring the Database for Synchronization*.

9. On the Standard Setup menu, enter the number for the language (Java or Groovy) that was used to write the server extension.

10. At this stage, you will be prompted to enter the fully qualified name of the Server SDK extension class that implements the JDBCSyncDestination API.

```
Enter the fully qualified name of the Java class that will implement
com.unboundid.directory.sdk.sync.api.JDBCSyncDestination:
com.unboundid.examples.oracle.ComplexJDBCSyncDestination
```

11. Next, the Synchronization Server prompts if you want to configure any user-defined arguments needed by the server extension. These are defined in the extension itself and the values are specified in the server configuration. If there are user-defined arguments, enter `yes`. Otherwise press **Enter** to accept the default (no) and continue. For this example, enter "yes" to configure the arguments for the script.

12. Next, the Synchronization Server prompts if you want to prepare the Source Endpoint server, which tests the connection to the directory server and tests that the Sync User account is accessible. Press Enter to accept the default (yes). For the Sync User account, it will return "Denied" as the account has not been written yet to the Directory Server at this time.

```
Testing connection to server1.example.com:1389 ..... Done
Testing 'cn=Sync User,cn=Root DNs,cn=config' access ..... Denied
```

13. Next, you will be prompted if you want to configure the Sync User account on the directory server. Press **Enter** to accept the default (yes). You will be prompted for the bind DN (e.g., cn=Directory Manager) and the bind DN password of the directory server so that you can configure the `cn=Sync User` account. The Synchronization Server creates the Sync User account, tests the base DN, and enables the change log.

```
Created 'cn=Sync User,cn=Root DNs,cn=config'
```

```
Verifying base DN 'dc=example,dc=com' ..... Done
Enabling cn=changelog .....
```

14. Next, you will be prompted to enter the maximum age of the change log entries. For this example, press **Enter** to accept the default (2d).

Step 2. Configuring the Sync Pipe and Sync Classes

In this section, we define the Sync Pipe and then create two Sync Classes. The first Sync Class is used to match the "accounts" objects. The second Sync Class is used to match the "group" objects. We'll set the basic Sync Class definitions and then add the attribute and DN maps in a later step.

To Configure the Sync Pipe and Sync Classes

1. Continuing from the previous session, enter a name for the Sync Pipe. Make sure the name is descriptive to identify it if you have more than one sync pipe configured. For example, enter `dssync-to-oraclesync`.

2. Next, you will be prompted if you would like to define one or more Sync Classes. Enter `yes`. We'll define the Accounts Sync Class, and then the Groups Sync Class in the next sections.

To Configure the Accounts Sync Class

1. Next, enter a name for the Sync Class. Make sure the name is descriptive to identify the sync class. For example, type `accounts_sync_class`.

2. At this stage, if you plan to restrict entries to specific subtrees, then enter one or more base DNs. For this example, press **Enter** to accept the default (no).

3. Next, you will be prompted to set an LDAP search filter. For this example, type yes to set up a filter and enter the filter `"(accountid=*)"`. Press **Enter** again to continue. This property sets the LDAP filters and returns all entries that match the search criteria to be included in the Sync Class. In this example, we want to specify that any entry with an accountID attribute be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.

4. Continuing from the previous example, on the Sync Class menu, you will be prompted if you want to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization. Press **Enter** to accept the default (all). We'll adjust these mappings in a later section.

5. Next, specify the operations that will be synchronized for the Sync Class. For this example, press **Enter** to accept the default (1, 2, 3) for creates, deletes, modifies.

To Configure the Groups Sync Class

For this current example, we need to configure another Sync Class to handle the Groups objectclass. The procedures are similar to that of the configuration steps for the `account_sync_class` Sync Class that were presented in the previous section.

1. On the Sync Class Management menu, enter a name for a new sync class. In this example, enter `groups_sync_class`.

2. At this stage, if you plan to restrict entries to specific subtrees, then enter one or more base DNs. Enter one or more base DNs. For this example, type `no`.

3. Next, you will be prompted to set an LDAP search filter. For this example, type `yes` to set up a filter and enter the filter `"(objectClass=groupOfUniqueNames)"`. Press **Enter** again to continue. This property sets the LDAP filters and returns all entries that match the `groupOfUniqueNames` attribute to be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.

4. Continuing from the previous example, on the Sync Class menu, you will be prompted if you want to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization. Press **Enter** to accept the default (all). We'll adjust these mappings in a later section.

5. Next, specify the operations that will be synchronized for the Sync Class. For this example, press **Enter** to accept the default (1, 2, 3) for creates, deletes, modifies.

6. At this point, you will see the Sync Class menu again asking you to enter the name of another Sync Class. Press **Enter** to continue.

7. Next, on the Default Sync Class Operations menu, press **Enter** to accept the default (1,2,3) for creates, deletes, and modifies. The Default Sync Class determines how all entries that do not match any other Sync Class are handled, including whether create, delete, and/or modify operations are synchronized.

8. Review the configuration, and then press **Enter** to write the configuration to the Synchronization Server. If you want to change any property, you can go back to the particular menu, or make the adjustments later using the dsconfig tool. If you decide to write the configuration to the Synchronization Server, press **Enter**, and then enter the connection properties for your Synchronization Server (bind DN, bind DN password).

```
>>>> Configuration Summary

  Sync Pipe: dssync-to-oraclesync

    Source: dssync
      Type: UnboundID Directory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: server1.example.com:1389

    Destination: oraclesync
      Type: Oracle Database
      Access Account: SyncUser
      Servers: dbsync-w2k8-vm-2:1521

    Sync Classes:
```

```
      accounts_sync_class
        Base DN:
        Filters: (accountID=*)
        DN Map: None
        Synchronized Attributes: -none-
        Operations: Creates,Deletes,Modifies

      groups_sync_class
        Base DN:
        Filters: (objectClass=groupOfUniqueNames)
        DN Map: None
        Synchronized Attributes: -none-
        Operations: Creates,Deletes,Modifies

      DEFAULT
        Operations: Creates,Deletes,Modifies

   w)  write configuration
   b)  back
   q)  quit  Enter

choice [w]:
```

9. The `create-sync-pipe-config` tool outputs the following messages. If you have to make any manual changes to the external servers, it will present them.

```
Creating External Servers ..... Done
Creating Endpoints ..... Done
Creating Sync Pipes ..... Done
Creating Attribute and DN Mappings ..... Done
Creating Sync Classes ..... Done

The following issues should be resolved before starting synchronization:

  Server 'dbsync-w2k8-vm-2:1521' needs manual preparation before starting
 synchronization.

  * You need to manually create the 'SyncUser' user account on this server and grant
    the proper privileges.

  You need to implement the following scripted adapter(s): com.unboundid.exam-
  ples.samples.ComplexJDBCSyncDestination.

  Refer to the product documentation for a recommended approach for initially
  bringing the two ends points into sync. Once this is done, you can enable
  real-time synchronization using the 'realtime-sync' tool.

Press RETURN to continue

See /UnboundID-Sync/logs/tools/create-sync-pipe-config.log for a detailed log of
this operation
```

Step 3. Fine-Tuning the Sync Classes

The Accounts and Groups Sync Classes require more fine-tuning as the DN and attributes maps need to be configured. Some additional properties are required for the example presented in this chapter.

To Fine-Tune the Accounts Sync Class

1. Start the `dsconfig` tool. Then, enter or select the LDAP (or LDAPS) connection parameters for the Synchronization Server.

```
$ bin/dsconfig
```

2. On the UnboundID Synchronization Server configuration console main menu, enter the number corresponding to Sync Class. On the Standard Objects menu, enter the corresponding number for Sync Class.

3. On the Sync Class Management menu, type 3 to view and edit an existing Sync Class.

4. Select or confirm that you are configuring a given Sync Pipe. Press Enter to continue.

5. Next, select the specific Sync Class that you want to modify. For this example, enter the number corresponding for the accounts sync class.

```
>>>> Select the Sync Class from the following list:

    1)  accounts_sync_class
    2)  DEFAULT
    3)  groups_sync_class

    b)  back
    q)  quit

Enter choice [b]: 1
```

6. On the Sync Class Properties menu, enter the number corresponding to the `description` property. For this example, enter "This Sync Class matches the site-user, guest, and administrator objectClasses." This step is optional but if you configure more than one Sync Class, you should add a general description describing the sync class's purpose.

```
>>>> Configure the properties of the Sync Class
 >>>> via creating 'account_sync_class' Sync Class

    1)   description                          "This Sync Class matches the site-user,
                                              guest, and administrator
                                              objectclasses."
    2)   evaluation-order-index               10
    3)   include-base-dn                      The location of the entry in the Sync
                                              Source is not taken into account when
                                              determining whether an entry is
                                              part of this Sync Class.
    4)   include-filter                       (accountID=*)
    5)   attribute-map                        No attribute map is used.
    6)   dn-map                               No dn map is used.
    7)   auto-mapped-source-attribute         all
    8)   excluded-auto-mapped-source-         No source attributes are excluded from
          attributes                            synchronization.
    9)   destination-correlation-attributes   dn
    10)  synchronize-creates                  true
    11)  synchronize-modifies                 true
    12)  synchronize-deletes                  true

    ?)   help
    f)   finish - create the new Sync Class
    a)   show advanced properties of the Sync Class
    d)   display the equivalent dsconfig arguments to create
          this object
    b)   back
    q)   quit

Enter choice [b]:
```

To Configure an Attribute Map

1. On the Sync Class Property menu, enter the corresponding to setting the attribute map. On the Attribute Map Property menu, enter 2 to add one or more values, and then, enter 1 to create a new attribute map.

2. Next, enter a name for the Attribute Map. Make sure the name is descriptive as you can typically have more than one attribute map in a Sync Class. For this example, enter Directory to DB Attr Map. Review the configuration on the Attribute Map Properties menu, and then enter `f` to save the configuration. We'll add the attribute mappings in a later section.

```
>>>> Configure the properties of the Attribute Map
 >>>> via creating 'Directory to DB Attr Map' Attribute Map

        Property     Value(s)
        ---------------------
    1)  description  -

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

Enter choice [b]: f
```

To Configure a DN Map

Next, we set up a DN Map from DNs in the form of *,ou=People,dc=example,dc=com and map it to a column/row value of "accountid={accountid}" in the database using the `dsconfig` command.

1. On the Sync Class Property menu, enter the number corresponding to the `dn-map` property. On the **DN Map Property** menu, enter 2 to add one or more values. Since there are no existing maps, enter 1 to create a new DN Map. Enter a name for the DN Map. For this example, enter `ubid_to_oracle_accounts_dn_map`. Review the configuration on the DN Map Properties menu, and then enter `f` to save the configuration.

2. Next, enter the name of the `from-dn-pattern` property on the source directory server. For example, enter "*,ou=People,dc=example,dc=com."

3. Next, enter the name of the `to-dn-pattern` property to which it will be mapped to the destination database server. For example, enter "accountid={accountid}."

4. On the DN Map Property menu, review the configuration, and then enter `f` to save and apply the changes.

```
>>>> Configure the properties of the DN Map
 >>>> via creating 'ubid_to_oracle_accounts_dn_map' DN Map
  >>>> via creating 'account_sync_class' Sync Class

         Property         Value(s)
         ----------------------------------------------
     1)  description      -
     2)  from-dn-pattern  "*,ou=People,dc=example,dc=com"
     3)  to-dn-pattern    accountid={accountid}

     ?)  help
     f)  finish - create the new DN Map
     d)  display the equivalent dsconfig arguments to create this object

     b)  back
     q)  quit

Enter choice [b]:f
```

5. On the DN Map Property menu, press **Enter** to use the value (`ubid_to_oracle_accounts_dn_map`) that you just entered.

To Configure the Ignore-Zero-Length-Values Property

1. On the Sync Class Property menu, type a to show the advanced properties. Then, enter the number corresponding to the `ignore-zero-length-values` property. This property ignores attribute changes that result in an empty (zero-length) value. Set the value to TRUE.

2. On the Sync Class Property menu, review the configuration, and type `f` to save and apply the changes. The advanced properties menu is displayed.

```
>>>> Configure the properties of the Sync Class
 >>>> via creating 'account_sync_class' Sync Class

    1)   description                            "This Sync Class matches the
                                                site-user. guest, and administrator
                                                objectclasses."
    2)   evaluation-order-index                 5
    3)   include-base-dn                        The location of the entry is in the
                                                Sync Source is not taken into
                                                account when determining whether an
                                                entry is part of this Sync Class.
    4)   include-filter                         (accountID=*)
    5)   attribute-map                          Directory to DB Attr Map
    6)   dn-map                                 ubid_to_oracle_accounts_dn_map
    7)   auto-mapped-source-attribute           -none-
    8)   excluded-auto-mapped-source-           No source attributes are excluded
          attributes                            from synchronization.
    9)   destination-correlation-attributes     accountID
    10)  destination-correlation-attributes-    -
          on-delete
    11)  synchronize-creates                    true
    12)  synchronize-modifies                   true
    13)  synchronize-deletes                    true
    14)  attribute-synchronization-mode         all-attributes
    15)  ignore-zero-length-values              true
    16)  replace-all-attr-values                true
    17)  modifies-as-creates                    false
    18)  creates-as-modifies                    false

    ?)   help
    f)   finish - create the new Sync Class
    a)   show advanced properties of the Sync Class
    d)   display the equivalent dsconfig arguments to
          create this object
    b)   back
    q)   quit

Enter choice [b]: f
```

   You have successfully configured the `account_sync_class` Sync Class.

To Fine-Tune the Groups Sync Class

For this current example, we need to configure another Sync Class to handle the Groups objectclass. The procedures are similar to that of the configuration steps for the `account_sync_class` Sync Class.

1. On the Sync Class Management menu, enter the number corresponding to View and Edit an Existing Sync Class, and then select `groups_sync_class`.

2. On the Sync Class Properties menu, configure the following properties:
   a) Set the description property to: "This Sync Class matches the Groups objectclass."
   b) Create and set the attribute map to: Directory to DB Groups Map

c) Create and set the DN map to: `ubid_to_oracle_groups_dn_map`. The equivalent `dsconfig` command is as follows:

```
$ bin/dsconfig create-dn-map \
  --map-name ubid_to_oracle_groups_dn_map \
  --set "from-dn-pattern:**" \
  --set "to-dn-pattern:name={cn}"
```

d) Set the `ignore-zero-length-values` property to: `true`

3. The specific property values for the Groups Sync Class can be seen below. When finished, review the configuration, and then enter `f` to save and apply the changes:

```
>>>> Configure the properties of the Sync Class
 >>>> via creating 'Groups Sync Class' Sync Class

        Property                                      Value(s)
        ---------------------------------------------------------------------------
   1)   description                           This Sync Class matches the Groups
                                             objectclass.
   2)   evaluation-order-index                10
   3)   include-base-dn                       The location of the entry in the Sync
                                             Source is not taken into account when
                                             determining whether an entry is part
                                             of this Sync Class.
   4)   include-filter                        (objectClass=groupOfUniqueNames)
   5)   attribute-map                         Directory to DB Groups Map
   6)   dn-map                                ubid_to_oracle_groups_dn_map
   7)   auto-mapped-source-attribute          -none-
   8)   excluded-auto-mapped-source-          No source attributes are excluded
         attributes                           from synchronization.
   9)   destination-correlation-attributes    dn
   10)  destination-correlation-attributes-   -
         on-delete
   11)  synchronize-creates                   true
   12)  synchronize-modifies                  true
   13)  synchronize-deletes                   true
   14)  ignore-zero-length-values             true
   15)  replace-all-attr-values               true
   16)  modifies-as-creates                   false
   17)  creates-as-modifies                   false

   ?)   help
   f)   finish - create the new Sync Class
   a)   hide advanced properties of the Sync Class
   d)   display the equivalent dsconfig arguments to
        create this object
   b)   back
   q)   quit

Enter choice [b]: f
```

4. On the Sync Class Management menu, enter `b` to back out of this menu to return to the UnboundID Synchronization Server configuration console main menu.

Step 4. Configuring the Attribute Mappings

In a previous step, the attribute maps were configured and added to each Sync Class (see *Configuring the Attribute Mappings* on page 141). Attribute maps are containers for attribute mappings that map the source attributes to similar or other attributes in the destination server. Based on the example schema, we want to configure the following Accounts and Group Table attributes on the system as follows:

Table 11: Attribute Mappings to Synchronize the Accounts Table

from-attribute (DS)	to-attribute (DB)
accountID	accountID
address	address
email	email
firstName	firstName
lastName	lastName
lastLogin	lastLogin
middleName	middleName
numLogins	numLogins
phone	phone

Table 12: Attribute Mappings to Synchronize the Group Table

from-attribute (DS)	to-attribute (DB)
cn	name
description	description
uniqueMember[1]	memberID

To Create the Attribute Mapping

1. On the UnboundID Synchronization Server configuration console main menu, enter the number corresponding to Attribute Mapping. On the Basic objects menu, enter the number corresponding to Attribute Mapping.

2. On the Attribute Map Management menu, enter the number corresponding to Create a New Attribute Mapping.

3. Select the Attribute Map that will be the container for this attribute mapping. For this example, enter the number corresponding to the Directory to DB Attr Map.

```
>>>> Select the Attribute Map from the following list:

    1)  Directory to DB Attr Map
    2)  Directory to DB Groups Map

    b)  back
    q)  quit

Enter choice [b]: 1
```

4. Next, select the type of attribute mapping that you want to create. In this example, enter the number corresponding to Direct Attribute Mapping.

5. Next, enter the name of the "to-attribute" to which the entry's attribute will be mapped on the destination database server. For this example, enter `accountID`.

6. Next, enter the name of the "from-attribute" from which it will be mapped to the "to-attribute" on the source directory server. For example, enter: `accountID`.

7. On the Direct Attribute Mapping Properties menu, review the configuration, and then type `f` to save the changes.

[1] DN attribute mapping

```
>>>> Configure the properties of the Direct Attribute Mapping
 >>>> via creating 'accountID' Direct Attribute Mapping

        Property        Value(s)
        -------------------------
    1)  to-attribute    accountID
    2)  description     -
    3)  from-attribute  accountID

    ?)  help
    f)  finish - create the new Direct Attribute Mapping
    a)  show advanced properties of the Direct Attribute Mapping
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]: f
```

8. Repeat steps 2–7 for the other attribute mappings.

- Or, you can use the `dsconfig` batch file feature to configure the attribute mappings at one time. Quit the `dsconfig` interactive session, create a text file, copy-and-paste the following `dsconfig` commands in the file, save the file as "attr-mappings.txt." Run the `dsconfig` command using the `-F` (or `--batch-file`) option. You must also use the `--no-prompt` option with the command. From the command line, run the `dsconfig` command and specify the batch file.

```
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name address --type direct --set from-attribute:address
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name email --type direct --set from-attribute:email
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name  firstName --type direct --set from-attribute:firstName
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name lastName --type direct --set from-attribute:lastName
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name lastLogin --type direct --set from-attribute:lastLogin
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name middleName --type direct --set from-attribute:middleName
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name numLogins --type direct --set from-attribute:numLogins
dsconfig create-attribute-mapping --map-name "Directory to DB Attr Map" \
--mapping-name phone --type direct --set from-attribute:phone

# Create the Group Attribute Mappings and assign them to the
# "Directory to DB Groups Map"
dsconfig create-attribute-mapping --map-name "Directory to DB Groups Map" \
--mapping-name description --type direct --set from-attribute:description

# Create the "Directory to Database Group Membership DN Map".
dsconfig create-dn-map \
--map-name "Directory to Database Group Membership DN Map" \
--set "from-dn-pattern:*,ou=people,dc=example,dc=com" \
--set "to-dn-pattern:{1}"
dsconfig create-attribute-mapping --map-name "Directory to DB Groups Map" \
--mapping-name memberID --type dn --set from-attribute:uniqueMember \
--set dn-map:"Directory to Database Group Membership DN Map"
dsconfig create-attribute-mapping --map-name "Directory to DB Groups Map" \
--mapping-name name --type direct --set from-attribute:cn

From the command line, run the following:

$ bin/dsconfig --port 7389 --bindPassword password \
--batch-file attr-mappings.txt --no-prompt
```

Step 5. Run the Resync Tool to Test the Configuration

The `resync` tool is used to test the sync configuration and connections. The tool has a `--dry-run` option that does not update the destination server but is convenient to test the configuration settings and report what is currently out of sync.

To Run Resync to Test the Configuration

- Run the `resync` command in "dry-run" mode to test the synchronization setup.

```
$ bin/resync --pipe-name dssync-to-oraclesync --dry-run
```

Step 6. Set the Startpoint in the Change Log

The `realtime-sync set-startpoint` command sets the starting point in the change log to tell the Synchronization Server where to start when the Sync Pipe is started. This command provides a way to avoid syncing all of the changes that have happened in the past.

To Set the Startpoint

- Run the `realtime-sync set-startpoint` command to mark the point to start tracking changes in the change tracking mechanism.

```
$ bin/realtime-sync set-startpoint --end-of-changelog \
--pipe-name dssync-to-oraclesync --port 389 --bindDN "cn=Directory Manager" \
--bindPassword password
```

Step 7. Run the Resync Tool to Populate Data at the Destination Endpoint

The `resync` tool is also used to populate a target server with data from the source.

To Run the Resync Tool to Populate Data onto a Target Server

- Run the `resync` command to populate data onto a newly configured target server. The Synchronization Server will make three passes to load data onto the server.

```
$ bin/resync --pipe-name dssync-to-oraclesync --numPasses 3
```

Step 8. Start the Sync Pipe

At this stage, we have configured everything necessary for the directory-to-database Sync Pipe. We only need to start it. Generally, it is preferable to use the `realtime-sync` tool to start and stop the Sync Pipes as well as start and stop the Sync configuration globally.

To Start the Sync Pipe

- Run the `realtime-sync` tool to start Sync Pipe.

```
$ bin/realtime-sync start --pipe-name dssync-to-oraclesync
```

Step 9. Debugging the Configuration

Typically, you will need to debug any problems after you run the `prepare-endpoint-server` and `resync` commands. There are a number of logging and tools options available when debugging the configuration as presented in the following sections.

Check the Status

- Run the `status` tool to verify the synchronization. You should check if the servers are connected and that changes are being detected. You can enter your bindPassword and have the system use your bindDN and port as defaults.

```
$ status --bindPassword password
```

- You can also restrict the status output to just list a single sync pipe using the `--pipe-name` option.

```
$ status --bindPassword password --pipe-name dssync-to-oraclesync
```

Check the Logs

- Increase the detail in the Sync log by changing the Sync Log Publisher handler's `logged-message-type` property to include: `change-applied-detailed`, `change-detected-detailed`, and `entry-mapping-details`.

```
$ dsconfig set-log-publisher-prop --publisher-name "File-Based Sync Logger" \
--set logged-message-type:change-applied-detailed \
--set logged-message-type:change-detected-detailed \
--set logged-message-type:change-failed-detailed \
--set logged-message-type:dropped-op-type-not-synchronized \
--set logged-message-type:dropped-out-of-scope \
--set logged-message-type:entry-mapping-details \
--set logged-message-type:no-change-needed
```

- Tail the `errors` log in the `logs` directory to locate any errors.

- Enable the debug logger (disabled by default), then rerun the `resync` command. You should disable the logger when no longer needed as it can impact performance.

```
# Enable the Debug Logger
dsconfig set-log-publisher-prop --publisher-name "File-Based Debug Logger" \

--set enabled:true

# Set the Debug Target and Verbosity Level
dsconfig create-debug-target --publisher-name "File-Based Debug Logger" \
--target-name com.unboundid.directory.sync.jdbc --set debug-level:verbose

# When finished with debugging, disable the logger
```

```
dsconfig set-log-publisher-prop --publisher-name "File-Based Debug Logger" \
--set enabled:false
```

- If your connections are working and the resync operation is working but you are seeing sync errors, tail the `sync-failed-ops` log. The problems could be in your attribute or DN maps.

Scripted Logging Methods

The ServerContext class provides several logging methods which can be used to generate log messages and/or alerts from the scripted JDBC layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These are described in the *Server SDK API Javadocs*.

Testing One Entry at a Time

Testing and debugging a configuration can be made more tractable if you test one entry at a time. When testing a directory-to-database sync configuration, the easiest way to do this is to use the `resync` tool's "`--sourceInputFile`" option, which allows you to specify a list of one or more DNs to sync.

When to Restart the Sync Pipe

- Make sure to restart the Sync Pipes after modifying a any extension code and rebuilding. You do not need to first run `realtime-sync stop`; running `realtime-sync start` will automatically re-start the pipe.

```
$ bin/realtime-sync start
```

- Because `resync` is a separate process and independently loads the server configuration, it is not necessary to restart the sync pipe.

Note: Any Synchronization Server configuration changes automatically restart the Sync Pipe. Extension implementation changes require a manual Sync Pipe restart.

Contact Your Support Provider

If you require assistance, your authorized support provider usually requests that you run the `bin/collect-support-data` command so that they can locate the source of any problems. The command generates a zip file that you can send to your support provider.

```
$ bin/collect-support-data --bindDN uid=admin,dc=example,dc=com \
--bindPassword password
```

General Tips When Syncing from a Database Source

When syncing from a database to a target endpoint server (directory server or RDBMS), remember to consider the following tips:

- **Identify Database Entry Types**. It is advisable to identify the different database entry types that will be synced. There are two things that you need to do:
  - Set the `database-entry-type` property on the JDBC Sync Source (this is required), and make sure the entry types are what the triggers are inserting into the change tracking mechanism.
  - Create a Sync Class per entry type, so that you can easily distinguish between them and have different mappings and sync rules set up for each one.
- **For each Sync Class**, do the following:
  - Make sure the `include-filter` property is set to match the entry type.
  - Create a specific Attribute Mapping for every database column that you want to be synced to a LDAP attribute; add this to a single Attribute Map and set it on the Sync Class. This way, the script will not have to know about the schema on the directory side.
  - Create a DN Map that recognizes the DNs generated by the script and map them to the correct location at the destination; set that on the Sync Class.
  - Remove the default value of "`-all-`" from the `auto-mapped-source-attribute` property on the Sync Class, and replace it with the value "objectClass". The object class for the fetched source entry is determined by the scripted layer. You do not want any values from the database automatically mapped to an attribute with the same name, which is why we set up explicit Attribute Mappings. The exception to this rule is the `objectclass` attribute, which we want to directly map for CREATE operations. If this is not done, an error is generated due to the lack of structural object class in the entry.
  - Change the `destination-correlation-attributes` property to contain the attributes that uniquely represent the database entries on the directory server destination. This will likely be something other than the default, which is "`dn`".
- **Avoid Bidirectional Loopback**. Make sure to set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization. This is important to make sure that changes are not looped back by the Synchronization Server.

Configuring the Database-to-Directory Sync Pipe

The setup procedure for a Sync Pipe from a database to the directory server is similar to that of the directory-to-database sync configuration. However, there are slight differences in terms of enabling or setting properties for bidirectional synchronization.

To display the additional features of the `dsconfig` command, the following procedure uses `dsconfig` in non-interactive mode to set up the Database-to-Directory Sync Pipe. You can run each command from the command line, in scripts, or in a batch file when setting up multiple configurations.

The procedures assume that you have already set up the directory-to-database Sync Pipe and that it is fully operational and connected. Remember to include the connection parameters (hostname, port, bindDN, and bindPassword) with each `dsconfig` command.

To Create the Database-to-Directory Sync Pipe

1. Run the `create-sync-pipe-config` tool to configure the Database-to-Directory Sync Pipe. The steps are similar to those presented in the previous sections.

2. Run the `resync` tool to test the configuration. When testing a database-to-directory Sync Pipe, you must specify the `--entryType` of the database table that is synchronized.

```
$ bin/resync --pipe-name oracle_to_ubid --entryType account --dry-run
```

3. Run the `realtime-sync` tool with the `set-startpoint` subcommand to mark the point to start tracking changes in the change tracking mechanism.

```
$ bin/realtime-sync set-startpoint --end-of-changelog --pipe-name oracle_to_ubid \
--port 389 --bindDN "cn=Directory Manager" --bindPassword password
```

4. Run the `resync` tool to populate data onto a newly configured target server. The Synchronization Server will make three passes to load data onto the server.

```
$ bin/resync --pipe-name oracle_to_ubid --numPasses 3 --entryType account
$ bin/resync --pipe-name oracle_to_ubid --numPasses 3 --entryType group
```

5. Run the `realtime-sync` tool to start Sync Pipe.

```
$ bin/realtime-sync start --pipe-name oracle_to_ubid
```

6. Troubleshoot the Sync Pipe as presented in *Step 9. Debugging the Configuration*.
You have successfully configured a bidirectional DBSync system.

Synchronizing a Specific List of Database Elements Using Resync

The `resync` command allows you to synchronize a specific set of database keys that are read from a JDBC Sync Source file using the `--sourceInputFile` option. The contents of the file are passed line-by-line into the `listAllEntries()` method of the JDBCSyncSource extension, which is used for the Sync Pipe. The method processes the input and returns DatabaseChangeRecord instances based on the input from the file.

To Synchronize a Specific List of Database Elements Using Resync

1. Create a file of JDBC Sync Source elements. The format of the file is up to the user, but it typically contains a list of primary keys or SQL queries. For example, create a file containing a list of primary keys and save it as sourceSQL.txt.

```
user.0
user.1
user.2
user.3
```

2. Run the `resync` command with the `--sourceInputFile` option to run on individual primary keys in the file.

```
$ bin/resync --pipe-name "dbsync-pipe" --sourceInputFile sourceSQL.txt
```

3. If you are targeting a specific type of database entry to search for, you can also use the `--entryType` option that matches one of the configured entry types in the JDBCSyncSource.

```
$ bin/resync --pipe-name "dbsync-pipe" \
--entryType account --sourceInputFile sourceSQL.txt
```

Chapter 6 Syncing Through Proxy Servers

The UnboundID Synchronization Server supports synchronization between directory servers and relational databases. Because most data centers deploy their directory servers in a proxied environment, the UnboundID Synchronization Server can also synchronize data through a proxy server in both load-balanced and entry-balancing deployments. The following types of proxy endpoints are supported:

- UnboundID Directory Proxy Server (version 3.x or later)
- Alcatel-Lucent 8661 Directory Proxy Servers (3.x or later)

The Sync-through-Proxy feature is only available for deployments in combination with a backend set of standalone or replicated UnboundID Directory Servers (version 3.x or later) or Alcatel-Lucent 8661 Directory Server (3.x or later).

This chapter presents the procedures to set up a Sync-through-Proxy deployment and provides some background information on how it works. Before setting up the Synchronization Server, review the section *Configuration Model* to understand the important components of the Synchronization Server. Also, review the Proxy Server Administration Guide for background information on the proxy server.

This chapter presents the following topics:

Topics:

- *Features*
- *How It Works*
- *About the Overall Sync-through-Proxy Configuration Process*
- *About the Sync-Through-Proxy Configuration Example*
- *Configuring the Example Source Proxy Deployment*
- *Configuring the Example Destination Proxy Deployment*
- *Indexing the LDAP Changelog*

Features

The UnboundID Synchronization Server (version 3.x) supports data synchronization through a proxy server from and to an endpoint consisting of the following:

- UnboundID Directory Proxy Server (version 3.x or later)
- Alcatel-Lucent 8661 Directory Proxy Server (version 3.x or later)

Each proxy server has a backend set of servers consisting of the following:

- UnboundID Directory Servers (version 3.x or later)
- Alcatel-Lucent 8661 Directory Server (version 3.x or later)

The servers have been updated with additional components to provide seamless synchronization through the proxy using the following features:

- Synchronization is fully supported for load-balanced and entry-balancing proxy server deployments.
- The Directory Server and the Directory Proxy Server provide a common interface to detect and retrieve changes to be synchronized, including failover to an alternate source server.
- The Directory Proxy Server provides a built-in server affinity mechanism to ensure change log searches are routed to the same directory server each time while it is online. This allows for more efficient processing compared to load-balancing the searches across the backend directory servers.
- The UnboundID Synchronization Server uses the same configuration procedures as any other endpoint setup. The proxy server's operations are largely transparent to the Synchronization Server.
- Proxy transformations are not supported. Any required transformations must be implemented in the Synchronization Server rather than the Proxy Server.

Note: If you are using the UnboundID Synchronization Server (version 3.x) with an earlier version of the UnboundID Directory Proxy Server and UnboundID Directory Server (versions 1.4.x, 2.2.x), you cannot run sync-through-proxy. The feature has not been backported to earlier versions.

How It Works

To handle data synchronization through a proxy server, the UnboundID Directory Server, UnboundID Directory Proxy Server, Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server, and the UnboundID Synchronization Server all have been updated with a new cn=changelog state management system that supports a token-based API and other components necessary for seamless data synchronization through the proxy. The tools have also been updated to handle these new components.

In a standard, non-proxied configuration, the Synchronization Server polls the source server for changes, determines if a change is necessary, and fetches the full entry from the source. Then, it finds the corresponding entry in the destination endpoint using flexible correlation rules and applies the minimal set of changes to bring any modified attributes into sync. The server fetches and compares the full entries to make sure it does not synchronize any stale data from the change log.

In a proxied environment, the Synchronization Server essentially does the same thing but transparently to the user, it passes the request through a proxy server to the backend set of directory servers. The Synchronization Server uses the highest priority proxy server designated in its endpoint server configuration and can quickly use other proxy servers in the event of a failover. Figure 18 shows an example deployment with two endpoints consisting of a proxy server deployment in front of the backend set of directory servers. Remember that you can have one endpoint consisting of UnboundID Directory Proxy Server and UnboundID Directory Servers while the other endpoint can be a directory server or RDBMS deployment (UnboundID Directory Server, Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Servers, Sun DSEE 6.x, 7.x, Sun Directory Server 5.2 patch 3 or higher, Microsoft Active Directory, Oracle 10g, 11g, or Microsoft SQL Server 2005, 2008).

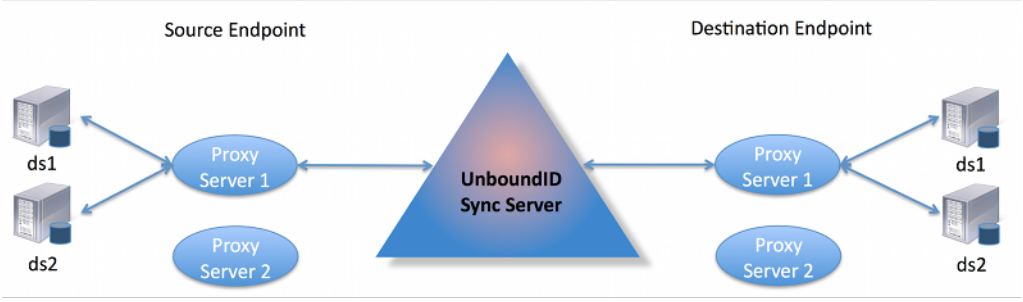


Figure 18: Sync-Through-Proxy

About the Get Changelog Batch Request and Get Server ID Controls

When the Synchronization Server runs a poll for any changes, it sends a Get Changelog Batch (GCB) Extended Request to the cn=changelog backend. The Get Changelog Batch looks for entries in the change log and asks for information on the server ID, change number, and replica state for each change. The Proxy Server routes the request to a directory server instance, which then returns a changed entry plus a token identifying the server ID, change number and replica state for each change. The proxy server then sends a Get Changelog Batch Response back to the Synchronization Server with this information. For entry-balancing deployments, the Directory Proxy Server must "re-package" the directory server tokens into its own proxy token to identify the specific data set. We will return to this a bit later.

To provide automatic server affinity in the proxied environment, the Synchronization Server uses the Get Server ID (GSID) Request Control together with the Get Changelog Batch (GCB) to identify the server ID of any fetched entry as illustrated in Figure 19. The first time that the Synchronization Server issues GCB request, it also issues a GSID Request Control to identify the specific server ID that is processing the extended request. The Directory Proxy Server routes the request to the directory server instance, and then returns a server ID in the response. Upon the next GCB request, the Synchronization Server sends a Route to Server (RTS) Request Control specifying the server instance to access again (in this example, server A) in this batch

session. It also issues a GSID Request Control to get an updated server ID in the event that the particular server (e.g., server A) is down. This method avoids round-robin server selection and provides more efficient overall change processing.

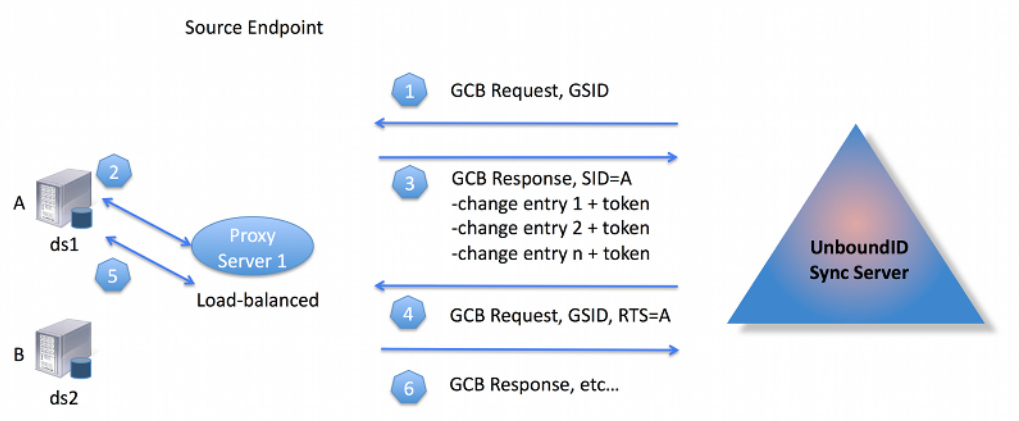


Figure 19: Get Changelog Batch Requests with Built-in Server Affinity

About the Directory Server and Directory Proxy Server Tokens

The Directory Server maintains a new change log database index to determine at what point to resume sending changes (corresponding to ADD, MODIFY, or DELETE operations) in its change log. While a simple stand-alone directory server can track its resume point by the last change number sent, it is more difficult for replicated servers or servers deployed in entry-balancing environments. In replicated environments, each replica has a different change number ordering in its change log as updates can come from a variety of sources: local write operations, changes from the other replication servers, or synchronized changes from other end-points. Figure 20 illustrates a simple chart of two example change logs in two replicated directory servers, server A and B. In the chart, A represents the replica identifier for a replicated subtree in Server A, and B represents the replica identifier for the same replicated subtree in server B. The replica identifiers with a hyphen ("-") mark any local, non-replicated but different changes. While the two replicas record all of the changes, you can see that the two change logs have two different change number orderings as updates come in at different times.

Server A			Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	A_{ri}	10	2001	B_{ri}	11
1002	-	-	2002	A_{ri}	10
1003	A_{ri}	15	2003	-	-
1004	B_{ri}	11	2004	B_{ri}	12
1005	B_{ri}	12	2005	A_{ri}	15

Figure 20: Different Change Number Order in Two Replicated Change Logs

To track the change log resume position, the Directory Server uses a change log database index to identify the latest change number position corresponding to the highest replicationCSN number for a given replica. This information is encapsulated in a directory server token and returned in the Get Changelog Batch Response control to the Directory Proxy Server. The token has the following format:

```
Directory Server Token: server ID, changeNumber, replicaState
```

For example, if the Proxy Server sends a request for any changed entries and the Directory Servers return the change number 1003 from server A and change number 2005 from server B, then each directory server token would contain the following information:

```
Directory Server Token A:
  serverID A, changeNumber 1003, replicaState {15(A)}

Directory Server Token B:
  serverID B, changeNumber 2005, replicaState {12(B), 15(A)}
```

Change Log Tracking in Entry-Balancing Deployments

Entry-balancing provides additional complexity in change log tracking in that a shared area of data can exist above the entry-balancing base DN in addition to each backend set having its own set of changes and tokens as mentioned previously. In Figure 21, the change logs of two servers are shown with server A belonging to an entry-balancing set 1 and server B belonging to an entry-balancing set 2. Shared areas that exist above the entry-balancing base DN are assumed to be replicated to all servers. Thus, SA represents the replica identifier for that shared area on server A and SB represents the replica identifier for the same area on server B.

Set 1 - Server A			Set 2 - Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	SA_{ri}	5	2001	SB_{ri}	10
1002	A_{ri}	10	2002	B_{ri}	20
1003	SB_{ri}	15	2003	SA_{ri}	5

Figure 21: Different Change Number Order in Two Replicated Change Logs

The Directory Proxy Server cannot simply pass a directory server token from the client to the backend directory server backend and back again as each directory server has its own set of changes and its tokens. Thus, in an entry-balancing deployment, the Proxy Server must maintain its own token mechanism that associates a directory server token (changeNumber, replicaIdentifier, replicaState) to a particular backend set.

```
Proxy Token:
backendSetID 1: ds-token 1 (changeNumber, replicaIdentifier, replicaState)
backendSetID 2: ds-token 2 (changeNumber, replicaIdentifier, replicaState)
```

For example, if the Directory Proxy Server returned change 1002 from server A and change 2002 from server CB, then the Proxy token would contain the following:

```
Proxy Token:
backendSetID 1: ds-token-1 {serverID A, changeNumber 1002, replicaState (5(SA), 15(A)}
backendSetID 2: ds-token-2 {serverID B, changeNumber 2002, replicaState (10(SB), 20(B)}
```

For each change entry returned by a backend, the Directory Proxy Server must also decide whether it is a duplicate of a change made to the backend set above the entry balancing base, since such changes appear in the change log across all backend sets. If the change is a duplicate, then it is discarded. Otherwise, any new change is returned with a new value of the proxy token.

About the Overall Sync-through-Proxy Configuration Process

The procedure to configure a Sync-through-Proxy system follows the basic procedures for a standard Sync configuration. The overall configuration process is as follows:

1. Set up your proxy server with its backend set of directory servers at one endpoint or both endpoints.
2. Download the Synchronization Server zip build, and unpack it to a directory of your choice.
3. From the server root directory of the Synchronization Server, run the `create-sync-pipe-config` command for your initial configuration. The command will interactively prompt you to input values necessary for your configuration.
4. Run the `prepare-external-server` command on the endpoint Directory Proxy Server instance and the backend set of directory servers. The Directory Proxy Server passes on a client request to the directory servers, which requires the cn=Sync User account be present on those servers for accessibility purposes. The LDAP Change Log is also enabled on the directory servers.
5. Run the `resync --dry-run` command to test the configuration settings.
6. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization. Note that you cannot use the `--change-number` option with a Sync-through-Proxy deployment but can use another option, such as `--end-of-changelog` or `--change-sequence-number` options.
7. Run the `resync` command to populate data on a target endpoint.
8. Start the Sync Pipes using the `realtime-sync start` command.
9. Monitor the synchronization server using the status commands and logs.

About the Sync-Through-Proxy Configuration Example

This section presents the steps to configure a sync-through-proxy network and uses an example configuration that has its two endpoints consisting of an UnboundID Directory Proxy Server with a backend set of UnboundID Directory Servers: both sets are replicated. The Directory Proxy Server uses an entry-balancing environment for the DN:`ou=People,dc=example,dc=com` and provides a subtree view for `dc=example,dc=com` in its client connection policy. For this example, we assume that communication will be over standard LDAP and that failover servers are not installed or designated in the Synchronization Server.

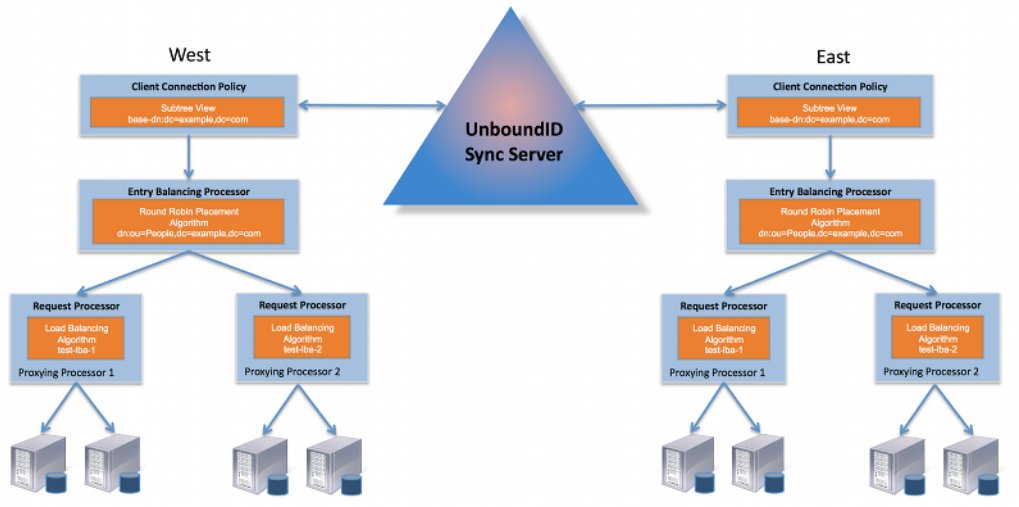


Figure 22: Example Sync-Through-Proxy Configuration

Configuring the Example Source Proxy Deployment

To configure the source proxy deployment, follow the procedures in the next two sections. The `--port` option defaults to 389, the `--bindDN` option defaults to "cn=Directory Manager", and the `--proxyBindDN` option defaults to "`cn=Proxy User,cn=Root DNs,cn=config`".

Configuring the Directory Servers

The following procedures present the basic `dsconfig` command-line instructions in non-interactive mode to set up this example's backend set of directory servers. The specific setup procedures may differ based on your particular environment. For more detailed background information, please review the *UnboundID Directory Server Administration Guide*.

To Configure the Directory Servers

1. To begin installing and configuring the directory servers, unzip the directory server file in a location of your choice.

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

2. If you plan to use SSL or StartTLS for communication, copy any keystore and truststore files to the `<server-root>/config` directory. For this example, we do not use SSL or StartTLS. All communication will be over standard LDAP.

3. If you have an existing schema file, copy the file to the `<server-root>/config/schema` directory.

4. Run the `setup` command from the root installation directory. Select your memory size options for your machine. For this example, create the base entry for the first directory server

instance in the first backend set at host name ldap-west-01.example.com. Set the maximum JVM heap size to 4 GB.

```
$ ./setup --cli --no-prompt --listenAddress ldap-west-01.example.com \
--ldapPort 389 --rootUserPassword password --baseDN dc=example,dc=com \
--aggressiveJVMTuning --maxHeapSize 4g --acceptLicense
```

5. Configure the directory server. Here you can configure your local DB indexes, virtual attributes, log files, password policies, SASL mechanisms and global configuration properties. Minimally, you must enable the change log database backend on your server instance, either from the command line or using a `dsconfig` batch file.

```
$ dsconfig --no-prompt set-backend-prop --backend-name changelog --set enabled:true
```

Note: If you do not plan to have the specific directory server instance participate in synchronization, you do not need to enable its change log.

6. Repeat steps 1–5 for the other instances. Make sure to specify the hostname and port for each server instance.

7. Import the dataset for the first backend set into the first server in the backend set. You must stop the server if it is running prior to the import.

```
$ bin/stop-ds
$ bin/import-ldif --backendID userRoot --ldifFile ../dataset.ldif
$ bin/start-ds
```

8. On the first server instance in the first backend set, configure replication between this server and the second server in the same backend set.

```
$ bin/dsreplication enable --host1 ldap-west-01.example.com \
  --port1 389 --bindDN1 "cn=Directory Manager" --bindPassword1 password \
  --replicationPort1 8989 --host2 ldap-west-02.example.com --port2 389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 password \
  --replicationPort2 9989 --adminUID admin --adminPassword admin \
  --baseDN dc=example,dc=com --no-prompt
```

9. Initialize the second server in the backend set with data from the first server in the backend set. This command can be run from either instance.

```
$ bin/dsreplication initialize --hostSource ldap-west-01.example.com \
  --portSource 389 --hostDestination ldap-west-02.example.com \
  --portDestination 389 --baseDN "dc=example,dc=com" --adminUID admin \
  --adminPassword admin --no-prompt
```

10. Run `dsreplication status` to check your replicas.

```
$ bin/dsreplication status --hostname ldap-west-01.example.com \
  --port 389 --adminPassword admin --no-prompt
```

11. Repeat steps 8 through 11 (import, enable replication, initialize replication, check status) for the second backend set.

To Configure the Directory Proxy Servers

The following procedures present the basic `dsconfig` command-line instructions to set up your proxy servers in non-interactive mode. The procedures configure the proxy servers from a bottom-up perspective: from defining the external servers to configuring the client-connection policy. If you are configuring the proxy servers for the first time, we recommend using the `create-initial-proxy-config` tool. The tool provides a command-line wizard presenting the interactive steps to configure your proxy server. For additional changes, you can use the dsconfig tool to fine-tune your proxy server. For more detailed background information, please review the *UnboundID Directory Proxy Server Administration Guide*.

1. To begin installing and configuring the directory servers, unzip the UnboundID Directory Proxy Server file in a location of your choice.

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

2. Run the `setup` command from the proxy server root installation directory. For this example, the default bind DN will be "cn=Directory Manager" and bind DN (or root user) password is set to "pxy-pwd." You can also use the `--aggressiveJVMTuning` with the `--maxHeapSize` options to set the amount of JVM memory for this application.

```
$ setup --cli --no-prompt --ldapPort 389 --rootUserPassword pxy-pwd \
--acceptLicense
```

3. From the Directory Proxy Server root directory, run the `prepare-external-server` command to set up the cn=Proxy User account and its privileges to give the proxy server access to the backend directory servers. After you press **Enter**, the command tests the connection to the server, creates the "cn=Proxy User" account, tests the connection to the account again, and checks the backend.

```
$ bin/prepare-external-server --no-prompt \
  --hostname ldap-west-01.example.com \
  --port 389 --bindDN "cn=Directory Manager" --bindPassword password \
  --proxyBindDN "cn=Proxy User,cn=Root DNs,cn=config" \
  --proxyBindPassword pass --baseDN "dc=example,dc=com"
```

4. Repeat step 3 for the other directory server instances in this example. Make sure to specify the specific hostname and port.

5. Next, run the `dsconfig` command to define the external servers and their types. The Directory Proxy Server communicates with these external servers through the cn=Proxy User account. Normally, you may want to set up any health checks and designate your server locations using this command. However, for this example, we use round-robin load-balancing algorithms, which do not require any health checks or locations to be specified.

```
$ bin/dsconfig --no-prompt create-external-server --server-name ldap-west-01 \
  --type "unboundid-ds" --set "server-host-name:ldap-west-01.example.com" \
  --set "server-port:389" --set "bind-dn:cn=Proxy User" \
  --set "password:password" --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd

$ bin/dsconfig --no-prompt create-external-server --server-name ldap-west-02 \
  --type "unboundid-ds" --set "server-host-name:ldap-west-02.example.com" \
  --set "server-port:389" --set "bind-dn:cn=Proxy User" \
  --set "password:password" --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server --server-name ldap-west-03 \
  --type "unboundid-ds" --set "server-host-name:ldap-west-03.example.com" \
  --set "server-port:389" --set "bind-dn:cn=Proxy User" \
  --set "password:password" --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd

$ bin/dsconfig --no-prompt create-external-server --server-name ldap-west-04 \
  --type "unboundid-ds" --set "server-host-name:ldap-west-04.example.com" \
  --set "server-port:389" --set "bind-dn:cn=Proxy User" \
  --set "password:password" --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

6. Next, create a load-balancing algorithm for each backend set. In this example, create one algorithm for the two replicated servers in the first backend set, and another for the two replicated servers in the second backend set.

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-1" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-01" \
  --set "backend-server:ldap-west-02" \
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd

$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-2" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-03"
  --set "backend-server:ldap-west-04"
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

7. Next, configure the proxying request processors. A request processor provides the logic to either process the operation directly, forward the request to another server, or hand off the request to another request processor. You will define two proxying request processors, one for each load-balanced directory server set.

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-1" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-1" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd

$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-2" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-2" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd
```

8. At this stage, we define an entry-balancing request processor. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. This request processor uses multiple proxying request processors to process operations for the various backend LDAP servers.

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "entry-balancing-processor" \
  --type "entry-balancing" \
  --set "entry-balancing-base-dn:ou=People,dc=example,dc=com" \

  --set "subordinate-request-processor:proxying-processor-1" \
  --set "subordinate-request-processor:proxying-processor-2" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd
```

9. Next, define the placement algorithm, which selects the server set to use for new add operations to create new entries. In this example, we define a placement algorithm with a

round-robin algorithm that forwards LDAP add requests to backends sets in a round-robin manner.

```
$ bin/dsconfig --no-prompt create-placement-algorithm \
  --processor-name "entry-balancing-processor" \
  --algorithm-name "round-robin-placement" \
  --set "enabled:true" --type "round-robin" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd
```

10. Define the subtree view that specifies the base DN for the entire deployment.

```
$ bin/dsconfig --no-prompt create-subtree-view \
  --view-name "test-view" \
  --set "base-dn:dc=example,dc=com" \
  --set "request-processor: entry-balancing-processor" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd
```

11. Finally, define a client connection policy that specifies how the client connects to the proxy server.

```
$ bin/dsconfig --no-prompt set-client-connection-policy-prop \
  --policy-name "default" --add "subtree-view:test-view" \
  --bindDN "cn=Directory Manager" --bindPassword pxy-pwd
```

You have successfully configured the first endpoint topology for the source servers.

Configuring the Example Destination Proxy Deployment

To configure the destination proxy deployment, follow the procedures in the previous two sections. A summary of the example configuration commands are listed in Table 6-1. The `--port` option defaults to 389, the `--bindDN` option defaults to "cn=Directory Manager", and the `--proxyBindDN` option defaults to "`cn=Proxy User,cn=Root DNs,cn=config`".

Table 13: Summary of Proxy Configuration Commands for the Source and Destination Deployments

Component	Source Proxy Topology	Destination Proxy Topology

Prepare External Servers	`prepare-external-server --no-prompt \`
	`--hostname "ldap-west-01.example.com" \`
	`--bindPassword "password" --base DN"`
	` dc=example,dc=com"`
	
	`prepare-external-server --no-prompt \`
	`--hostname "ldap-west-02.example.com" \`
	`--bindPassword "password" \`
	`--baseDN" dc=example,dc=com"`
	
	`prepare-external-server --no-prompt \`
	`--hostname "ldap-west-03.example.com"\`
	`--bindPassword "password" \`
	`--baseDN" dc=example,dc=com"`
	
	`prepare-external-server --no-prompt \`
	`--hostname "ldap-west-04.example.com" \`
	`--bindPassword "password" \`
	`--baseDN" dc=example,dc=com"`
		`prepare-external-server \`
		`--no-prompt \`
		`--hostname "ldap-east-01.example.com"`
		` \`
		`--bindPassword "password" \`
		`--base DN" dc=example,dc=com"`
		
		`prepare-external-server \`
		`--no-prompt \`
		`--hostname "ldap-east-02.example.com"`
		` \`
		`--bindPassword "password" \`
		`--baseDN" dc=example,dc=com"`
		
		`prepare-external-server \`
		`--no-prompt \`
		`--hostname "ldap-east-03.example.com"`
		` \`
		`--bindPassword "password" \`
		`--base DN" dc=example,dc=com"`
		
		`prepare-external-server \`
		`--no-prompt \`
		`--hostname "ldap-east-04.example.com"`
		` \`
		`--bindPassword "password" \`
		`--baseDN" dc=example,dc=com"`

External Servers	`dsconfig create-external-server \`
	`--server-name: "ldap-west-01" \`
	`--type "unboundid-ds" \`
	`--set "server-host-name:ldap-`
	`west-01.example.com" \`
	`--set "server-port:389" \`
		`dsconfig create-external-server \`
		`--server-name: "ldap-east-01" \`
		`--type "unboundid-ds"`
		`--set "server-host-name:ldap-`
		`east-01.example.com" \`
		`--set "server-port:389" \`

Component	Source Proxy Topology	Destination Proxy Topology

	`--set "bind-dn:cn=Proxy User" \`
	`--set "password:password"`
	
	`dsconfig create-external-server \`
	`--server-name: "ldap-west-02" \`
	`--type "unboundid-ds" \`
	`--set "server-host-name:ldap-west-02.example.com" \`
	`--set "server-port:389" \`
	`--set "bind-dn:cn=Proxy User" \`
	`--set "password:password"`
	
	`dsconfig create-external-server \`
	`--server-name: "ldap-west-03" \`
	`--type "unboundid-ds" \`
	`--set "server-host-name:ldap-west-03.example.com" \`
	`--set "server-port:389" \`
	`--set "bind-dn:cn=Proxy User" \`
	`--set "password:password"`
	
	`dsconfig create-external-server \`
	`--server-name: "ldap-west-04" \`
	`--type "unboundid-ds" \`
	`--set "server-host-name:ldap-west-04.example.com" \`
	`--set "server-port:389" \`
	`--set "bind-dn:cn=Proxy User" \`
	`--set "password:password"`
		`--set "bind-dn:cn=Proxy User" \`
		`--set "password:password"`
		
		`dsconfig create-external-server \`
		`--server-name: "ldap-east-02" \`
		`--type "unboundid-ds" \`
		`--set "server-host-name:ldap-east-02.example.com" \`
		`--set "server-port:389" \`
		`--set "bind-dn:cn=Proxy User" \`
		`--set "password:password"`
		
		`dsconfig create-external-server \`
		`--server-name: "ldap-east-03" \`
		`--type "unboundid-ds" \`
		`--set "server-host-name:ldap-east-03.example.com" \`
		`--set "server-port:389" \`
		`--set "bind-dn:cn=Proxy User" \`
		`--set "password:password"`
		
		`dsconfig create-external-server \`
		`--server-name: "ldap-east-04" \`
		`--type "unboundid-ds" \`
		`--set "server-host-name:ldap-east-04.example.com" \`
		`--set "server-port:389" \`
		`--set "bind-dn:cn=Proxy User" \`
		`--set "password:password"`

Load-Balancing Algorithm	`dsconfig create-load-balancing-algorithm \`
	`--algorithm-name "test-lba-1" \`
	`--type "round-robin" \`
	`--set "enabled:true" \`
	`--set "backend-server: ldap-west-01 \`
	`--set "backend-server: ldap-west-02 \`
	`--set "use-location:false"`
	
	`dsconfig create-load-balancing-algorithm \`
	`--algorithm-name "test-lba-2" \`
	`--type "round-robin" \`
	`--set "enabled:true" \`
	`--set "backend-server: ldap-west-03 \`
	`--set "backend-server: ldap-west-04 \`
	`--set "use-location:false"`
		`dsconfig create-load-balancing-algorithm \`
		`--algorithm-name "test-lba-1" \`
		`--type "round-robin" \`
		`--set "enabled:true" \`
		`--set "backend-server: ldap-east-01 \`
		`--set "backend-server: ldap-east-02 \`
		`--set "use-location:false"`
		
		`dsconfig create-load-balancing-algorithm \`
		`--algorithm-name "test-lba-2" \`
		`--type "round-robin" \`
		`--set "enabled:true" \`
		`--set "backend-server: ldap-east-03 \`
		`--set "backend-server: ldap-east-04 \`
		`--set "use-location:false"`

Request Processors (load-balancing)	`dsconfig dsconfig create-request-processor \`
	`--processor-name "proxying-processor-1" \`
	`--type "proxying" \`
	`--set "load-balancing-algorithm:test-lba-1"`
	
	`dsconfig create-request-processor \`
	`--processor-name "proxying-processor-2" \`
	`--type "proxying" \`
	`--set "load-balancing-algorithm:test-lba-2`
	
	`dsconfig create-request-processor \`
	`--processor-name "entry-balancing-processor" \`
	`--type "entry-balancing" \`
	`--set "entry-balancing-base-dn:ou=People,dc=example,dc=com" \`
	`--set "subordinate-request-processor:proxying-processor-1" \`
	`--set "subordinate-request-processor:proxying-processor-2"`
		Same as source

Placement Algorithm	`dsconfig create-placement-algorithm \`
	`--processor-name "entry-balancing-processor" \`
	`--algorithm-name "round-robin-placement" \`
	`--set "enabled:true" \`
	`--type "round-robin"`
		Same as source

Subtree View	`dsconfig create-subtree-view \`
	`--view-name: "test-view" \`
	`--set "base-dn:dc=example,dc=com" \`
	`--set "request-processor:entry-balancing-rocessor"`
		Same as source

Client Connection Policy	`dsconfig set-client-connection-policy-prop \`
	`--policy-name: "default" \`
	`--add "subtree-view:test-view"`
		Same as source

To Configure the Synchronization Server

At this stage, the UnboundID Directory Proxy Server and its backend set of UnboundID Directory Server instances should be configured and fully functional for each endpoint, which is labelled as ldap-west and ldap-east in this example.

1. Download the UnboundID Synchronization ZIP file. Unzip the file in a directory of your choice.

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

2. If this is the first time that you are installing the Synchronization server on this machine, you must ensure that you have JDK1.6 update 25. Set the JAVA_HOME environment variable and your PATH or CLASSPATH variables accordingly.

3. From the Synchronization Server root directory, run the `setup` tool. For this example, the default bindDN will be "cn=Directory Manager" and the rootUser Password (or root user) is set to "password".You can also use the `--aggressiveJVMTuning` with the `--maxHeapSize` options to set the amount of JVM memory for this application.

   ```
   $ setup --no-prompt --ldapPort 389 --rootUserPassword password --acceptLicense
   ```

4. From the Synchronization Server root directory, run the `create-sync-pipe-config` tool, and then, press **Enter** to continue.

   ```
   $ bin/create-sync-pipe-config
   ```

5. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.

6. On the Synchronization Mode menu, press **Enter** to select Standard mode. A standard Mode Sync Pipe will fetch the full entries from both the source and destination and compare them to produce the minimal set of changes to bring the destination into sync. A notification mode Sync Pipe will skip the fetch and compare phases of processing and simply notify the destination that a change has happened and provide it with the details of the change. Notifications are currently only supported from UnboundID and Alcatel-Lucent Directory or Proxy Servers 3.0.3 or later.

7. On the Synchronization Directory menu, enter the number associated with the type of synchronization you want to configure: 1 for One-Way, 2 for bidirectional. For this example, type 1 for one-way, which will require that you configure one Sync Pipes (e.g., "proxy 1 to proxy 2").

8. Next, you will be prompted to configure the first endpoint server, which will be the first Directory Proxy Server topology. On the First Endpoint Type menu, enter the number for the type of backend datastore for the first endpoint. In this example, type the number corresponding to the UnboundID Proxy Server.

   ```
   >>>> First Endpoint Type

   Enter the type of data store for the first endpoint:

       1)  UnboundID Directory Server
       2)  UnboundID Proxy Server
   ```

```
    3)  Alcatel-Lucent Directory Server
    4)  Alcatel-Lucent Proxy Server
    5)  Sun Directory Server
    6)  Microsoft Active Directory
    7)  Microsoft SQL Server
    8)  Oracle Database
    9)  Custom JDBC

    b)  back
    q)  quit

Enter choice [1]: 2
```

9. Next, enter a descriptive name for the first endpoint. For this example, use "UnboundID Proxy 1".

10. Next, enter the base DN where the Synchronization Server can search for the entries on the first endpoint server. For this example, press **Enter** to accept the default, `dc=example,dc=com`.

11. Specify the type of security when communicating with the endpoint server. For this example, select **None**.

12. Enter the hostname and port of the endpoint server. The Synchronization Server will automatically test the connection to the endpoint server. Repeat the step if you are configuring another server for failover.

13. Next, enter the Sync User account that will be used to access the endpoint server (i.e., proxy server 1). Enter `cn=Sync User,cn=Root DNs,cn=config`, then, enter a password for the account.

14. At this point, you have defined the first endpoint deployment using the Proxy Server (e.g., ldap-west). Repeat steps 8-13 to define the second proxy deployment (e.g., ldap-east) on the Synchronization Server.

15. At this point, you will be prompted to "prepare" the endpoint servers in the topology. The endpoint servers here refer to the proxy servers in this example. This step ensures that the Sync User account is present on each server and that it has the proper privileges to allow communication between the Synchronization Server and the proxy servers. In addition to preparing the proxy server, the Synchronization Server must also prepare the backend set of directory servers as the proxy server passes through the authorization to access these servers. If they have not been prepared, you will see the following messages to invoke the commands prior to starting synchronization. Also note that each endpoint is a source and a destination in a bidirectional sync network; therefore, you must use `--isSource` and `--isDestination` options. If you are configuring a one-way Sync Pipe, you must specify `--isSource` for the first endpoint.

```
Discovering additional servers that require preparation .....

Server ldap-west-01.example.com:389 requires preparation. Before
starting synchronization you must invoke the following command,
substituting the correct password for [password]:
     prepare-endpoint-server --hostname ldap-west-01.example.com --port 389 \
          --baseDN dc=example,dc=com --isSource --isDestination \
          --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
          --syncServerBindPassword "[password]"

Server ldap-west-02.example.com:389 requires preparation.  Before
starting synchronization you must invoke the following command,
substituting the correct password for [password]:
     prepare-endpoint-server --hostname ldap-west-02.example.com --port 389 \
```

```
          --baseDN dc=example,dc=com --isSource --isDestination \
          --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
          --syncServerBindPassword "[password]"

Server ldap-west-03.example.com:389 requires preparation.  Before
starting synchronization you must invoke the following command,
substituting the correct password for [password]:
     prepare-endpoint-server --hostname ldap-west-03.example.com --port 389 \
          --baseDN dc=example,dc=com --isSource --isDestination \
          --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
          --syncServerBindPassword "[password]"

Discovering additional servers that require preparation ..... Done
```

16. Next, repeat step 15 to prepare the second endpoint server (i.e., in this example, the second proxy server). Again, if you have not prepared the underlying directory servers (e.g., ldap-east-01, ldap-east-02, ldap-east-03), you will need to run the commands prior to starting synchronization.

17. Define the Sync Pipe from proxy 1 to proxy 2. First, enter a descriptive name for the Sync Pipe. In this example, accept the default "UnboundID Proxy 1 to UnboundID Proxy 2."

18. Next, if you want to customize on a per-entry basis how attributes get synchronized, you must define one or more sync classes. Type yes if you have specific attribute or DN mappings, create a sync class for the special cases, and use default sync class for all other mappings. For this example, press **Enter** to accept the default (no).

19. For the default Sync Class Operations, specify the operations that will be synchronized for the default sync class. For this example, accept the default ([1,2,3]) for Creates, Deletes, and Modifies.

20. Finally, review the configuration settings, and then accept the default (write configuration) to the Synchronization Server. The Synchronization Server writes your configuration settings to a file, sync-pipe-cfg.txt, so that you can apply these configurations to other failover Synchronization Servers if necessary. Connect to the Synchronization Server so that the server will be updated with your settings.

To Confirm the Proxy Server and Use-Changelog-Batch-Request Properties

1. If you did not use the create-sync-pipe-config tool to create your Sync configuration, there are two properties that you need to verify on each endpoint: proxy-server and use-changelog-batch-request. The proxy-server property should specify the name of the proxy server, while the use-changelog-batch-request should be set to true on the Sync Source only. The use-changelog-batch-request is not available on the Destination end-point. Remember to add the connection parameters to your Synchronization Server (host- name, port, bind DN, and bind password). The following commands check the properties on a Sync Source.
   On the Sync Source:

   ```
   $ bin/dsconfig --no-prompt \
   get-sync-source-prop \
   --source-name "UnboundID Proxy 1" \
   --property "proxy-server" \
   --property "use-changelog-batch-request"
   ```

   On the Sync Destination:

   ```
   $ bin/dsconfig --no-prompt \
   get-sync-source-prop \
   ```

```
--source-name "UnboundID Proxy 2" \
--property "proxy-server"
```

2. From the server root directory, run the dsconfig command to set a flag indicating that the endpoints are proxy servers. Remember to add the connection parameters for the Synchronization Server (hostname, port, bind DN, and bind password) with the following commands:

```
$ bin/dsconfig --no-prompt \
set-sync-source-prop \
--source-name "UnboundID Proxy 1" \
--set proxy-server:ldap-west-01 \
--set use-changelog-batch-request:true

$ bin/dsconfig --no-prompt \
set-sync-source-prop \
--source-name "UnboundID Proxy 2" \
--set proxy-server:ldap-east-01
```

To Run Prepare-External-Server on the Backend Set of Directory Servers

1. From the server root directory, run the `prepare-external-server` command on each of directory server instances in the first endpoint topology that you want to have participate in synchronization.

```
$ prepare-endpoint-server \
--hostname ldap-west-01.example.com --port 389 \
--baseDN dc=example,dc=com --isSource  \
--syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
--syncServerBindPassword "password"

$ prepare-endpoint-server \
--hostname ldap-west-02.example.com --port 389 \
--baseDN dc=example,dc=com --isSource \
--syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
--syncServerBindPassword "password"

$ prepare-endpoint-server \
--hostname ldap-west-03.example.com --port 389 \
--baseDN dc=example,dc=com --isDestination \
--syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
--syncServerBindPassword "password"

$ prepare-endpoint-server \
--hostname ldap-west-04.example.com --port 389 \
--baseDN dc=example,dc=com --isDestination \
--syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
--syncServerBindPassword "password"
```

2. Repeat the previous step on the other endpoint topology (e.g., ldap-east).

To Test and Start the Configuration

1. Run the `resync --dry-run` command to test the configuration settings. We recommend running it for each sync pipe, debug any issues, then run the command again for the other sync pipe.

```
$ bin/resync --pipe-name "UnboundID Proxy 1 to UnboundID Proxy 2" --dry-run
```

2. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.

```
$ realtime-sync set-startpoint --end-of-changelog \
--pipe-name "UnboundID Proxy 1 to UnboundID Proxy 2" --port 389 \
--bindDN "cn=Directory Manager" \
--bindPassword password
```

Note: For Sync-through-Proxy deployments, you cannot use the `--change-number` option with the `realtime-sync set-startpoint` command as the Synchronization server cannot retrieve specific change numbers from the backend set of directory servers. You can use the `--change-sequence-number`, `--end-of-changelog` or the other options available for the tool.

3. Run the `resync` command to populate data on the endpoint destination server if necessary.

```
$ bin/resync --pipe-name "UnboundID Proxy 1 to UnboundID Proxy 2" --numPasses 3
```

4. Start the Sync Pipe using the `realtime-sync start` command.

```
$ bin/realtime-sync start --pipe-name "UnboundID Proxy 1 to UnboundID Proxy 2"
```

5. Monitor the synchronization server using the status commands and logs.
   You have successfully configured a Sync-through-Proxy deployment.

Indexing the LDAP Changelog

The UnboundID Directory Server (3.0 or later) and the Alcatel-Lucent 8661 Directory Server (3.0 or later) both support attribute indexing in the Changelog Backend to allow Get Changelog Batch requests to filter results that include only changes involving specific attributes. For example, if you are running a Sync-through-Proxy configuration in an entry-balanced deployment, the Synchronization Server sends a Get Changelog Batch request to the Proxy Server, which will send out individual Get Changelog Batch requests to each backend server. Each directory server that receives a request must iterate over the whole range of changelog entries and then match entries based on search criteria for inclusion in the batch. The majority of this processing involves determining whether a changelog entry includes changes to a particular attribute or set of attributes, or not. Using changelog indexing, client applications can dramatically speed up throughput when targeting these specific attributes.

Administrators can configure attribute indexing using the `index-include-attribute` and `index-exclude-attribute` properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

To determine if the directory server supports this feature, administrators can view the Root DSE for the following entry:

```
supportedFeatures: 1.3.6.1.4.1.30221.2.12.3
```

To Configure Changelog Indexing

This procedure assumes that the backend set of directory servers is comprised of either the UnboundID Directory Server (3.0 or later) or the Alcatel-Lucent 8661 Directory Server (3.0 or later), which is fronted by an UnboundID Directory Proxy Server (3.0 or later) or an Alcatel-Lucent 8661 Directory Proxy Server (3.0 or later). You do not need to configure the Directory Proxy Server as it passes the GetChangelogBatch requests to the backend directory servers.

1. On all source Directory Servers, enable changelog indexing for the particular attributes that will be synchronized. Use the combination of the `index-include-attribute` and `index-exclude-attribute` properties. The following example specifies that all user attributes ("`index-include-attribute:*`") be indexed in the changelog, except the `description` and `location` attributes ("`index-exclude-attribute:description`" and "`index-exclude-attribute:location`").

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "index-include-attribute:*" \
  --set "index-exclude-attribute:description \
  --set "index-exclude-attribute:location
```

Note: There is practically no performance and disk consumption penalty when using "`index-include-attribute:*`" with a combination of `index-exclude-attribute` properties versus explicitly defining each attribute using `index-include-attribute` alone. The only cautionary note about using "`index-include-attribute:*`" is to be careful that unnecessary attributes get indexed.

2. On the Synchronization Server, go to the Sync Class Management menu, and configure the `auto-map-source-attributes` property to specify the explicit mappings for the attributes that need to be synchronized. Note that you cannot use the `-all-` value for the `auto-map-source-attributes` property as this will not take advantage of changelog indexing. You must explicitly list out the attributes that should be auto-mapped.

Note:

The Synchronization Server will write a `NOTICE` message to the error log when the Sync Pipe first starts up, indicating whether the server is using changelog indexing or not.

```
[30/Mar/2012:13:21:36.781 -0500] category=SYNC severity=NOTICE
msgID=1894187256 msg="Sync Pipe 'TestPipe' is not using changelog
indexing on the source server"
```

The message appears under the following conditions: 1) if the source server supports changelog indexing, 2) if the attribute mappings are set up in such a way that will allow the Synchronization Server to use changelog indexing (i.e., using specific attribute mappings and not setting the `auto-map-source-attributes` property to `-all-`).

Chapter 7 Configuring Notification Mode

The UnboundID Synchronization Server supports a notification synchronization mode that transmits change notifications on a source endpoint to third-party destination applications. As is the case with synchronization running in standard mode, notifications can be filtered based on the type of entry that was changed, the specific attributes that were changed, and the type of change (ADD, MODIFY, DELETE). The Synchronization Server can send a notification to arbitrary endpoints by using a custom server extension based on the UnboundID Server SDK.

One deployment example is the implementation of a 3GPP-compliant Subscriber Data Management system. The Synchronization Server-based system generates SOAP XML-formatted push notifications over HTTP and transmits them to front-end applications whenever a change in the backend subscriber database occurs. In this example, the Synchronization Server processes the subscriber changes using a custom extension based on the UnboundID Server SDK. The custom extension and other third-party libraries manage the connection and protocol logic necessary to send the notifications to its front-end applications.

This chapter presents the background information and procedures to set up a notification mode system:

Topics:

- *About Notification Mode*
- *About the Notification Mode Configuration*
- *About the Server SDK and LDAP SDK*
- *Important Design Questions*
- *Implementing the Custom Server Extension*
- *Configuring the Notification Sync Pipe*
- *Access Control Filtering on the Sync Pipe*
- *Contact Your Support Provider*

About Notification Mode

The UnboundID Synchronization Server, version 3.1.0 or later, supports two modes of synchronization: standard and notification. Standard Mode is the default mode used to synchronize changes between its two endpoints. In standard mode, the Synchronization Server polls the directory server's LDAP Change Log for all create, modify, and delete operations on any entry. It fetches the full entries from both the source and destination endpoints and compares them to produce the minimal set of changes needed to bring the destination server in sync with the source server. The Synchronization Server completes the process by updating the destination endpoint with the necessary changes.

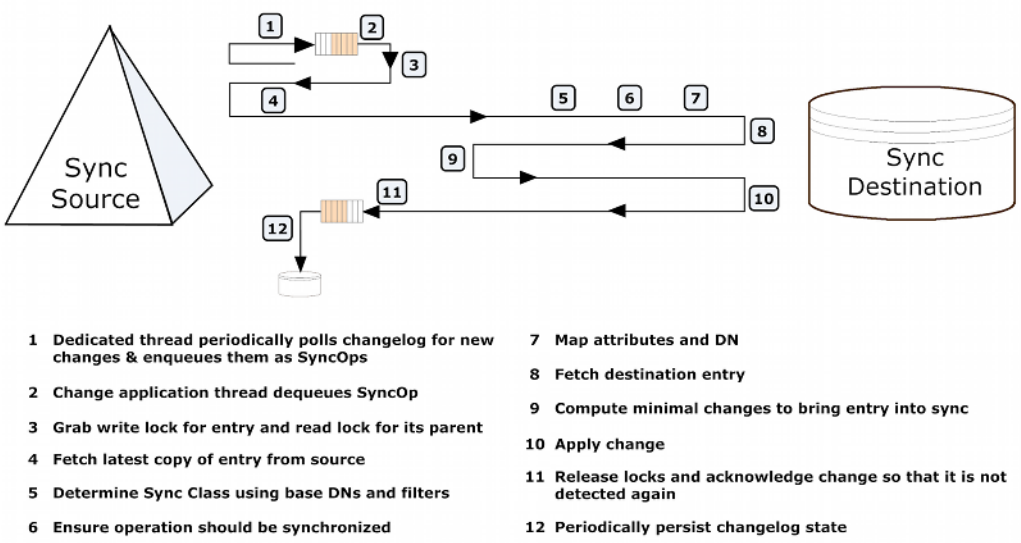


Figure 23: Standard Mode Synchronization Change Flow

The Synchronization Server provides another way to process changes called Notification Mode that polls the directory server's LDAP Change Log for changes on any entry but skips the fetch and compare phases of processing. Instead, the Sync Destination is notified of the change regardless of the current state of that entry at the source or destination. The Synchronization Server accesses state information on the change log to reconstruct the before-and-after values of any modified attribute (for example, for MODIFY change operation types). It passes in the change information to a custom server extension based on the UnboundID Server SDK.

Third-party libraries can be employed to customize the notification message to an output format required by the client application or service. For example, the server extension can use a third-party XML parsing library to convert the change notifications to a SOAP XML format. Notification mode can only be used with an UnboundID Directory Server, Alcatel-Lucent 8661 Directory Server, UnboundID Directory Proxy Server, or Alcatel-Lucent 8661 Directory Proxy Server as the source endpoint.

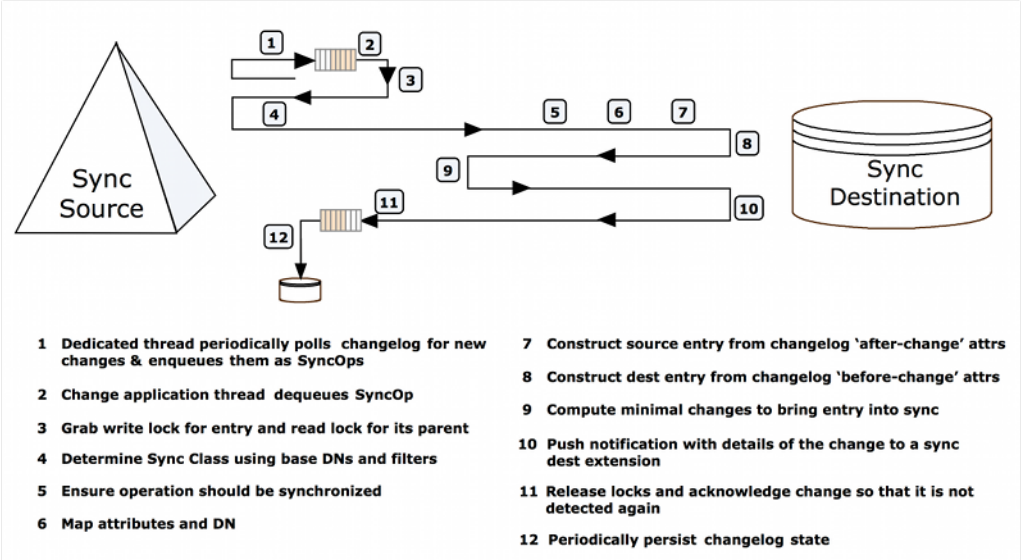


Figure 24: Notification Mode Synchronization Change Flow

Note: The Synchronization Server can use notification mode with any type of endpoint; therefore, it is not an absolute requirement to have a custom server extension in your system. For example, it is possible to set up a notification sync pipe between two LDAP server endpoints although it is not a practical production deployment scenario.

Notification Mode Architecture

Notification mode requires a one-way directional sync pipe from a source endpoint topology to a target client application. The Synchronization Engine detects the changes in the directory server's LDAP Change Log, filters the results specified in the Sync Classes, applies any DN and attribute mappings, then reconstructs the change information from the change log attributes. The server extension picks up the notification arguments from the SyncOperation interface (part of the Server SDK) and converts the data to the desired output format. The server extension establishes the connections and protocol logic to push the notification information to the client applications or services.

Note: The UnboundID Server SDK ships with documentation and examples on how to create a directory server extenstion to support notification mode.

For a given entry, the Synchronization Server sends notifications in the order that the changes occurred in the change log even if a modified attribute has been overwritten by a later change. For example, if an entry's telephoneNumber attribute is changed three times, three notifications will be sent in the order they appeared in the change log.

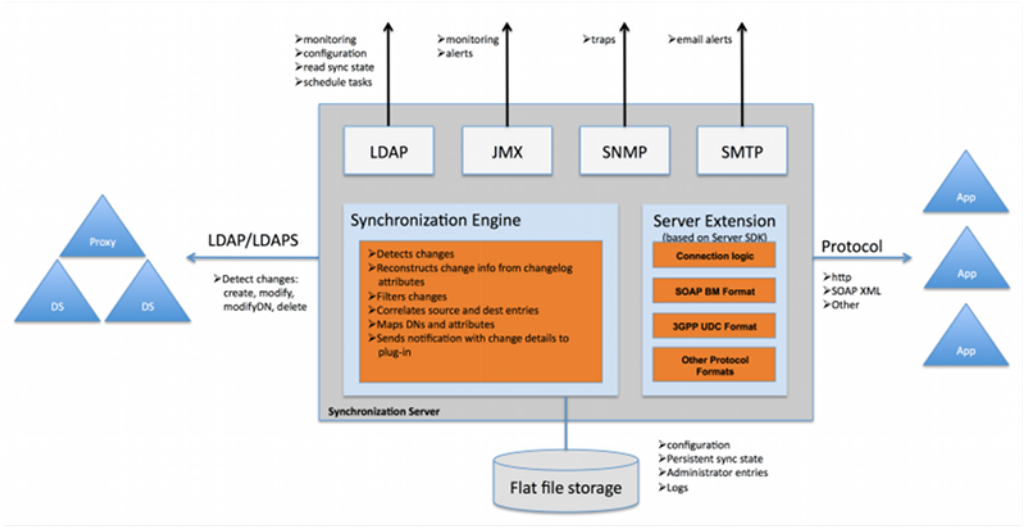


Figure 25: Notification Mode Architecture

Sync Source Requirements

In Notification Mode, a separate Sync Pipe is required for each client application that should receive a notification. The Sync Sources must consist of one or more instances of the following directory or proxy servers with the UnboundID Synchronization Server (version 3.1.0 or later):

- UnboundID Directory Server (version 3.0.5 or later)
- UnboundID Directory Proxy Server (version 3.0.5 or later)
- Alcatel-Lucent 8661 Directory Server (version 3.0.5 or later)
- Alcatel-Lucent 8661 Directory Proxy Server (version 3.0.5 or later)

The Sync Destination can be of any type.

Note: While the UnboundID Directory Proxy Server and Alcatel-Lucent 8661 Directory Proxy Server can front other vendor's directory servers, such as Active Directory and Sun DSEE, for processing LDAP operations, the UnboundID Synchronization Server cannot synchronize changes from these sources through the Directory Proxy Server. Synchronizing changes directly from Active Directory and Sun DSEE is supported but not with notification mode.

Failover Capabilities

To ensure high availability in the source backend directory servers, administrators should set up replication on the directory servers to ensure data consistency among the servers. Additionally, administrators can front the backend directory server set with a proxy server to redirect traffic should connection to the primary server fail. It is also necessary to use a proxy server for synchronizing changes in an entry-balancing environment. Once the primary directory server is online, it assumes control with no information loss as its state information is kept across the backend directory servers.

For destination failovers, the connection retry logic to the applications must be implemented in the server extension, which will then use the Sync Pipe's advanced property settings to retry any failed operations. Note that there is a difference between a connection retry and an operation retry. An extension should not retry operations since the Synchronization Server does so automatically. But the custom server extension is responsible for re-establishing connections to a destination that has gone down and/or failing over to an alternate server. The server extension can also be designed to trigger its own error-handling code during the failed operation.

For Synchronization Server failovers, the secondary Synchronization Servers will be at or slightly behind the state where the primary server initiated a failover. Both primary and secondary Synchronization Servers track the last failed acknowledgement, so once the primary server fails over to a secondary server, the secondary server will not miss a change.

Note: If failover is a concern between Synchronization Servers, you can change the `sync-failover-polling-interval` property from 5000 ms to a smaller value. This will result in a quicker failover but will marginly increase traffic between the two Synchronization Servers. Use `dsconfig` to access the property on the Global Sync Configuration menu.

Standard Administration and Monitoring Capabilities

The Notification mode is a configuration setting on the Sync Pipe. All of the operations, administration, and management (OA&M) functions available in standard mode, such as monitoring, (LDAP, JMX, SNMP), alerts (JMX, SNMP, SMTP), and extensive logging features remain the same for notification mode.

Notification Sync Pipe Change Flow

Figure 26 shows the change flow that occurs in the notification sync pipe. Although not pictured, the changes are processed in parallel using multi-threading, which increases throughput and offsets network latency. A single change-detection thread is dedicated to pull in batches of change log entries and queue them internally. Multi-threaded sync pipes allow the Synchronization Server to process multiple notifications in parallel in the same manner as synchronizing changes in standard mode. To guarantee consistency, the Synchronization Server's internal locking mechanisms ensure the following properties:

- Changes to the same entry will be processed in the same order that they appear in the change log.
- Changes to parent entries will be processed before changes to its children.
- Changes to entries with same RDN value are handled sequentially.

The number of concurrent threads is configurable on the Sync Pipe using the `num-worker-threads` property in the Synchronization Server. This configuration property determines how many operations can be processed in parallel. It can be set to "1" for those applications that require strict serial processing. In general, we recommend that the single-threading strategy be avoided to ensure that throughput and performance are not limited.

Apart from the threading model, one important aspect of the synchronization flow is that noti- fication mode does not fetch the full source and destination entries in comparison to standard mode. The Synchronization Server reconstructs the entries from specialized change log attributes that record the before-and-after values and entry-key attributes for each modification. See *LDAP Change Log Features Required for Notifications* for more information.

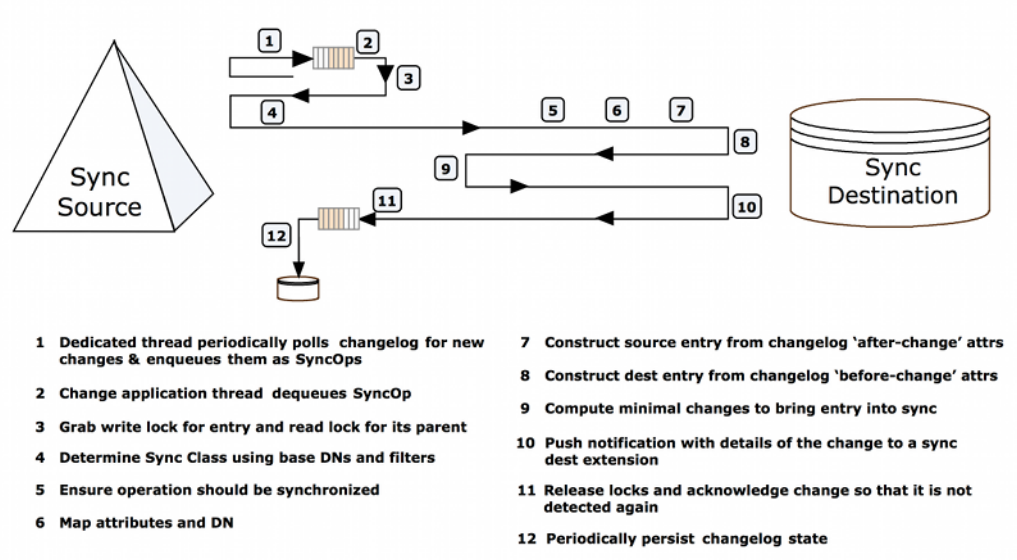


Figure 26: Notification Sync Pipe Change Flow

About the Notification Mode Configuration

The Synchronization Server supports notification mode with the following components.

Create-Sync-Pipe-Config

The `create-sync-pipe-config` tool supports the configuration of notification mode. Any pre-existing sync sources can be read from the local configuration (in the `config.ldif` file), so that redefining your sync sources is unnecessary if your topology is using a topology of servers consisting of the UnboundID Directory Server (3.0.5 or later) or the Alcatel-Lucent 8661 Directory Server (3.0.5 or later) and possibly fronted by an UnboundID Directory Proxy Server or an Alcatel-Lucent Directory Proxy Server.

No Resync

The `resync` function is disabled on a Sync Pipe in notification mode as its functionality is not supported in this implementation. Notification mode views the directory server's change log as a rolling set of data that pushes out change notifications to its target application. The notion of bringing the destination endpoints in-sync with the source endpoint only applies to standard synchronization mode.

LDAP Change Log Features Required for Notifications

As of version 3.0.3, the UnboundID Directory Server and the Alcatel-Lucent 8661 Directory Server have expanded their configuration to support notification mode with the addition of two new advanced global change log properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

The properties are enabled and configured during the `create-sync-pipe-config` configuration process on the Synchronization Server. The properties can also be enabled on the directory servers using the dsconfig advanced properties setting on the Backend->Changelog menu and are described in the following sections:

changelog-include-key-attribute

The `changelog-include-key-attribute` property specifies one or more attributes that should always be included in the change log entry. The purpose of this property is to specify those attributes needed to correlate entries between the source and destination, such as uid, employeeNumber, mail, etc. The other reason these properties are needed is for evaluating any filters in the Sync Class. For example, if notifications are only sent for user entries, and the Sync Class included the filter "(objectclass=people)", then the objectclass attribute must be configured as a `changelog-include-key-attribute` so that the Sync Pipe can evaluate the inclusion criteria when processing the change. In standard mode, values needed in the filter are read from the entry itself after it is fetched instead of from the changelog entry. Note also that these attributes are always included in a change log entry, also called a change record, regardless if they have changed or not.

The `changelog-include-key-attribute` property causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the change log entry. This applies for all change types. On a DELETE operation, the values are from the entry before it was deleted. The key values are recorded on every change and override any settings configured in the `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry-exclude-attribute` properties in the directory server changelog (see the UnboundID Directory Server Configuration Reference for more information).

Normal LDAP to LDAP synchronization topologies typically use "dn" as a correlation attribute. If you use "dn" as a correlation attribute only, you do not need to set the `changelog- include-key-attribute` property. However, if you require another attribute for correlation (e.g., uid, subscriberNumber, customerNumber, etc.), then you must set this property by specifying it during the configuration process (see *Configuring the Notification Sync Pipe*).

Table 14: LDAP Change Log Attributes: ds-changelog-entry-key-attr-values

LDAP Change Log Attributes	Description
ds-changelog-entry-key-attr-values	Stores the attribute that is always included in a change log entry on every change for correlation purposes. In addition to regular attributes, you can also specify virtual and operational attributes as your entry keys. To view an example, see the *UnboundID Directory Server Administration Guide*.

changelog-max-before-after-values

The `changelog-max-before-after-values` property specifies a single value greater than zero that sets the maximum number of before-and-after values (default: 200) that should be stored for any changed attribute in the change log. Also, when enabled, it will add the `ds-changelog-before-values` and `ds-changelog-after-values` attributes to any change record that contains changes (i.e., only Modify and ModifyDN).

The main purpose of the `changelog-max-before-after-values` property is to ensure that you do not store an excessively large number of before-and-after changes for multi-valued attributes in an change log entry. In most cases, the directory server's schema defines a multi-valued attribute to be unlimited in an entry. For example, if you have a group entry whose member attribute references 10000 entries, you may not want to record all of the attributes if a new member is added. The property safeguards against this scenario.

If either the `ds-changelog-before-values` or the `ds-changelog-after-values` attributes exceed the count set in the `changelog-max-before-after-values` property, the attribute values are no longer stored in a change record but its attribute name and number is stored in the `ds-changelog-attr-exceeded-max-values-count` attribute, which appears in the change record.

In addition to this property, you should also set the use-reversible-form property to "TRUE". This guarantees that sufficient information is stored in the change log for all operation types to be able to replay the operations at the destination. The `create-sync-pipe-config` tool sets up both of these properties if you choose to let it prepare the servers.

To summarize, the `changelog-max-before-after-values` property sets up the following change log attributes, seen in Table 7-2:

Table 15: LDAP Change Log Attributes: changelog-max-before-after-values

LDAP Change Log Attributes	Description
ds-changelog-before-values	Captures all "before" values of a changed attribute. It will store up to the specified value in the `changelog-max-before-after-values` property (default 200).
ds-changelog-after-values	Captures all "after" values of a changed attribute. It will store up to the specified value in the `changelog-max-before-after-values` property (default 200).
ds-changelog-attr-exceeded-max-values-count	Stores the attribute names and number of before/after values on the change log entry after the maximum number of values (set by the changelog-max-before-after-values property) has been exceeded. This is a multi-valued attribute whose format is: `attr=attributeName,beforeCount=200,afterCount=201` where "attributeName" is the name of the attribute and the "beforeCount" and "afterCount" are the total number of values for that attribute before and after the change, respectively. In either case (before or after the change) if the number of values is exceeding the maximum, then those values will not be stored.

LDAP Change Log for Notifications and Standard Mode

Both notification and standard mode sync pipes can consume the same LDAP Change Log without affecting the other. Standard mode polls the change record in the change log for any modifications, fetches the full entries on the source and the destination, and then compares

them for the specific changes. Notification mode gets the before-and-after values of a changed attribute to reconstruct an entry and bypasses the fetch-and-compare phase. Both can consume the same LDAP Change Log with no performance loss or conflicts.

Note: If your configuration obtains the change log through the proxy server, the contents of the change log will not change as it is being read from the change logs on the directory server backend.

About the Server SDK and LDAP SDK

The Server SDK and the LDAP SDK for Java have been updated to support the features required for notification mode. The specific changes are highlighted in the sections below. For detailed information, see the javadoc for the respective SDK.

The Synchronization Server engine processes the notification and makes it available to a ServerSDK extension, which can be written in Java or Groovy. Similar to database synchronization, place the custom server extension in the `<server-root>/lib/groovy-scripted-extensions` folder (for Groovy-based extensions) or the jar file in the `<server-root>/lib/extensions` folder (for Java-based extensions) prior to configuring the Synchronization Server for notification mode. Groovy scripts are compiled and loaded at runtime.

Server SDK Updates

To support notification mode, the Server SDK has been updated with a new extension type, SyncDestination, which is a generic endpoint used to synchronize with any type of client application. The architecture makes no assumptions about the type of output and processing required for the client applications as they are handled by the server extension. This generic extension type can also be used for standard synchronization mode.

An important interface that your server extension will use is the SyncOperation interface. The interface represents a single synchronized change from the Sync Source to the Sync Destination. The same SyncOperation object exists from when a change is detected all the way through when the change is applied at the destination. See the Server SDK Javadoc for detailed information.

Some methods that are implemented by the server extension are summarized as follows (for detailed information and examples, see the Server SDK Javadoc and the provided examples):

Table 16: SyncDestination Class

SyncDestination Class	Description
defineConfigArguments	Defines any configuration arguments needed for your extension. For example, this method can be used to configure the URL of a remote host to send a notification to. These arguments can then be used with the dsconfig tool in interactive and non-interactive (scripted) modes, and the web console.
initializeSyncDestination	Defines a life cycle method to initialize the Sync Destination.
createEntry	Creates the full destination entry, corresponding to the LDAP entry that is passed in.
modifyEntry	Modifies an entry on the destination, corresponding to the LDAP entry that is passed in.

SyncDestination Class	Description
deleteEntry	Deletes a full entry (in LDAP form) from the destination endpoint, corresponding to the source Entry that is passed in.
fetchEntry	This method exists in the API to provide a generic solution that works for standard sync mode. It is not needed in a notification mode deployment.
finalizeSyncDestination	Defines a life cycle method to finalize the Sync Pipe when it shuts down.
getCurrentEndpointURL	Returns the URL or path identifying the destination endpoint to which this extension is transmitting data.

LDAP SDK Updates

To support notification mode, the LDAP SDK for Java has been updated to support the before- and-after attributes in the change log. The LDAP SDK provides a new class, `UnboundIDChangelogEntry` (in the `com.unboundid.ldap.sdk.unboundidds` package) that has high level methods to work with the `ds-changelog-before-value`, `ds-changelog-after-values`, and `ds-changelog-entry-key-attr-values` attributes. The class is part of the commercial edition of the LDAP SDK for Java and is installed automatically with the Synchronization Server. For detailed information and examples, see the *LDAP SDK Javadoc*.

Important Design Questions

Before you begin implementing and configuring your sync pipe in notification mode, you should consider the following design questions:

- What is the interface to the client applications?
- What type of connection logic is required?
- How will the extension handle timeouts and connection failures?
- What are the failover scenarios?
- What data needs to be included in the change log?
- How long do the change log entries need to be available?
- What are the scalability requirements for the system?
- What attributes should be used for correlation?
- What should happen with each type of change?
- What mappings must be implemented?

Implementing the Custom Server Extension

Notification mode relies heavily on the server extension code to process and transmit the change using the required protocol and data formats needed for the client applications. You can create the extension using the UnboundID Server SDK, which provides the APIs to develop code for any destination endpoint type. The Server SDK's documentation (javadoc and examples) is delivered with the Server SDK build in zip format. The SDK provides all of the necessary classes to extend the functionality of the Synchronization Server without code changes to the core product. Once the server extension is in place, you can use other third-party libraries to transform the notification to any desired output format.

General Tips When Implementing Your Extension

When configuring a Sync Pipe in notification mode, you should be aware of the following recommendations:

- **Use the manage-extension Tool**. You can use the `manage-extension` tool in the `bin` directory (UNIX/LINUX) or `bat` directory (Windows) to install or update the extension. See the *Managing Extensions* section for more information.
- **Review the Server SDK Package**. The Server SDK comes with its own documentation and examples that show how to build and deploy a java or groovy extension. Note that to deploy a java extension, you must stop the server, copy the jar file to the `lib/extensions` folder, and then re-start the server. For Groovy extensions, copy the script to `lib/groovy-scripted-extensions` folder, and then re-start the sync pipe, which will reload the scripted extensions. You do not have to stop and re-start the server for Groovy extensions.
- **Connection & Protocol Logic**. The Server SDK-based extension must manage the notification connection and protocol logic to the client applications.
- **Implementing Extensions**. We recommend doing incremental development of your extension code or scripts. Start by testing the create methods, then the delete methods, and then the modify methods for each entry type. Write some code, test it, make adjustments, and repeat again. Then update the configuration. Finally, package the extensions for deployment. You can also increase the sync logging levels to see more details about what is happening with your extensions.
- **Use the SyncOperation Type**. The `SyncOperation` class encapsulates everything to do with a given change. Objects of this type are used in all of the Sync SDK extensions. The `SyncOperation` class has been updated to include new methods for support notification mode (see the Server SDK Javadoc for the `SyncOperation` class for information on the full set of methods):

Table 17: SyncOperation

Method	Description
getDestinationEntryBeforeChange()	Gets the destination entry before the change.
getDestinationEntryAfterChange()	Gets the destination entry after the change.
isModifyDN()	Determines if the changes is a MODIFY DN operation without looking at the change entry.
getChangelogEntry()	Gets the original change log entry to retrieve any attributes from it. This is the original source change before any mappings.
getSyncClass()	Gets a specific sync class and its components.
getType()	Returns the type of this SyncOperation.
logError()	Logs an error message to the synchronization log for this change.
logInfo()	Logs an information message to the synchronization log for this change.

- **Use the EndpointException Type**. The Sync Destination type throws a new sync exception type called `EndpointException`. This extends a standard Java exception, so that you can wrap other types of throwables and provide your own exceptions. There is also logic to handle LDAP exceptions, using the LDAP SDK, and wrap them into an `EndpointException`.

- **About the PostStep result codes**. The `EndpointException` class throws uses `PostStep` result codes that are returned in the server extension:

Table 18: PostStep

PostStep Result Codes	Description
retry_operation_limited	If set, this will retry a failed attempt up to the limit set by max_operation_attempts. Finally, it will be logged as failed.
retry_operation_unlimited	Retry the operation an unlimited number of times until a success, abort, or retried_operation_limited. This should only be used when the destination endpoint is unavailable.
abort_operation	Aborts the current operation without any additional processing.

- **Use the ServerContext class for logging**. The ServerContext class provides several logging methods which can be used to generate log messages and/or alerts from the scripted layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These are described in the Server SDK API Javadocs. Logging related to an individual `SyncOperation` should be done with the `SyncOperation#logInfo` and `SyncOperation#logError` methods.

- **Diagnosing Script Errors**. When a Groovy extension does not behave as expected, first look in the error log for stack traces. If you see `classLoader` errors, the script could be in the wrong location or does not have the correct package. Groovy code errors are very good at highlighting the line number where the error occurs. Groovy checks for errors at run-time. Business logic errors must be systematically found by testing each operation (Creates Modifies, Deletes). Make sure logger levels are set high enough to debug.

Configuring the Notification Sync Pipe

The following procedure shows the interactive steps to set up a one-way Sync Pipe with an UnboundID Directory Server as the Sync Source and a generic sync destination. The procedure uses the `create-sync-pipe-config` tool in interactive command-line mode, which shows the configuration steps in a top-down flow from Sync Pipe. Many of the configuration steps shown in this section are similar to those seen in previous chapters. The section only highlights the differences for configuring a Sync Pipe in notification mode.

The procedure is broken out into sections for easy access and is based on the interactive prompts that the `create-sync-pipe-config` tool will present. The instructions assume that the user has the proper root user or admin privileges to make configuration changes. Once you have configured the sync pipe, then you can fine-tune the configuration later using the `dsconfig` utility.

General Tips When Configuring Your Sync Classes

When configuring a sync class for a Sync Pipe in notification mode, you should be aware of the following recommendations:

- **Exclude Operational Attributes**. You may want to exclude any operational attributes from syncing to the destination so that its before-and-after values are not recorded in the change log. For example, the following attributes can be excluded: `creatorsName`,

`createTimeStamp`, `ds-create-time`, `ds-entry-unique-id`, `ds-update-time`, `modifiersName`, and `modifyTimeStamp`.

There are three methods to accomplish this depending on your directory server version. It is preferable to filter the changes at the change log level over making the changes in the Sync Class to avoid extra configuration settings:

- For version 3.0.3 of the UnboundID Directory Server or the Alcatel-Lucent 8661 Directory Server, use the directory server's `changelog-exclude-attribute` property to specify each operational attribute that you want to exclude in the synchronization process. You can set the configuration using the `dsconfig` tool on the directory server Change Log Backend menu. For example, set `changelog-exclude-attribute:modifiersName`.
- For version 3.1.0 of the UnboundID Directory Server or the Alcatel-Lucent 8661 Directory Server, use the directory server's `changelog-exclude-attribute` property with the special character, "+". For example, to exclude all operational attributes, set "`change-log-exclude-attribute:+`".
- On version 3.1.0 of the UnboundID Synchronization Server, you can configure a Sync Class that sets the `excluded-auto-mapped-source-attributes` property to each operational attribute that you want excluded from the synchronization process.

- **Consider Advanced Properties on the Sync Class**. The Synchronization Server has some advanced properties that you might want to consider using for your notifications sync topology depending on your design objectives.
  - **destination-create-only-attribute**. This property sets the attributes that you want to include on CREATE operations only but never want to modify. For example, you would specify objectclass as an attribute that you do not want to modify on the destination.
  - **replace-all-attr-values**. This property specifies whether to use the ADD and DELETE modification types (reversible), or the REPLACE modification type (non-reversible) for modifications to destination entries. If set to true, REPLACE will be used; otherwise, ADD and DELETE of individual attribute values will be used.
- **Consider Changelog Indexing**. If you target specific attributes and require higher performance throughput, consider implementing changelog indexing. See the *Syncing Through Proxy Servers* chapter for more information.

Step 1. Creating the Notification Sync Pipe

The initial configuration steps show how to set up a single Sync Pipe from a directory server instance to a generic sync destination client using the `create-sync-pipe-config` tool in interactive mode. The `create-sync-pipe-config` tool prompts the user for input and leads you through the configuration steps in a wizard-like mode. The procedure will show how to set up and configure the Sync Pipe, External Servers, and Sync Classes.

Optionally, you can run the `create-sync-pipe-config` tool with the server offline and apply the configuration later.

Before You Begin

1. Place any third-party libraries used in your application in the `<server-root>/lib/extensions` folder.
2. Implement your server extension and place it into the appropriate directory before starting any Sync Pipe that uses this endpoint. Custom endpoints require a Server SDK extension in order to interface with the target data store. The general location for the extensions should be the following:

   - Java extensions: <server-root>/lib/extensions
   - Groovy extensions: <server-root>/lib/groovy-scripted-extensions

   Because the Synchronization Server must reference the fully qualified class name for the extension, it must reside in the appropriate sub-directories. For example, if the extension is in the com.unboundid.sdk.examples.groovy package, then it must be placed in the `<server-root>/lib/groovy-scripted-extensions/com/unboundid/sdk/examples/groovy` folder.

To Create a Sync Pipe in Notification Mode

1. Start the Synchronization Server.

   ```
   $ bin/start-sync-server
   ```

2. Run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

3. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.

4. On the Synchronization Mode menu, select the option for notification mode. A standard Mode Sync Pipe will fetch the full entries from both the source and destination and compare them to produce the minimal set of changes to bring the destination into sync. A notification mode Sync Pipe skips the "fetch and compare" phases of processing and simply notify the destination that a change has happened and provide it with the details of the change. Notifications are currently only supported from UnboundID and Alcatel-Lucent Directory or Directory Proxy Servers 3.0.5 or later.

5. On the Synchronization Directory menu, enter the option to create a one-way Sync Pipe in notification mode from directory to a generic client application.

To Configure the Sync Source

1. On the Source Endpoint Type menu, enter the number for the sync source corresponding to the type of source external server. For this example, enter the option to select the UnboundID Directory Server.

2. If any pre-existing Sync Sources are present in the local server (stored in `config.ldif`), the tool asks if you want to select the sources listed. Enter the number corresponding to the Sync Source listed, or type n to create a new sync source.

3. Next, if you are creating a new Sync Source, you will be prompted to enter a name for the Source Endpoint. Enter a descriptive name for the Sync Source. For example, ds1.

4. Next, enter the base DN for the directory server, which is used as the base for LDAP searches. For example, enter dc=example,dc=com, and then press **Enter** again to return to the menu. If you enter more than one base DN, make sure the DNs do not overlap.

5. On the Server Security menu, select the type of secure communication that the Synchronization Server will use with the endpoint server instances. Select either 1) None; 2) SSL; or 3) StartTLS. For this example, select the default (None).

6. Next, enter the host and port of the first Source Endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The Synchronization Server will contact this first server if it is available, then contact the next highest priority server if the first server is unavailable, etc. After you have entered the host and port, the Synchronization Server tests that a connection can be established.

7. On the Synchronization Server User Account menu, enter the DN of the sync user account and create a password for this account. The Sync User account allows the Synchronization Server to access the source endpoint server. By default, the Sync User account is placed at `cn=Sync User,cn=Root DNs,cn=config`. Press **Enter** to accept the default configuration.

To Configure the Destination Endpoint Server

1. Next, on the Destination Endpoint Type menu, select the type of datastore on the endpoint server. In this example, select the option for Custom.

2. Next, you will be prompted to enter a name for the Destination Endpoint. Enter a descriptive name for the Sync Destination. For example, "Custom Destination".

3. On the Notifications Setup menu, select the language (Java or Groovy) that was used to write the server extension.

4. At this stage, you will be prompted to enter the fully qualified name of the Server SDK extension that implements the abstract class. If you wrote your extension in Java, the extension should reside in the `/lib/extensions` directory.

```
Enter the fully qualified name of the Java class that
will implement com.unboundid.directory.sdk.sync.api.SyncDestination:
com.unboundid.sdk.examples.ExampleSyncDestination
```

   - If you wrote your extension in Groovy, the script should reside in the `/lib/groovy-scripted-extensions` directory and is verified by the Synchronization Server.

```
Enter the fully qualified name of the Groovy class that will implement
com.unboundid.directory.sdk.sync.scripting.ScriptedSyncDestination:
com.unboundid.sdk.examples.groovy.ExampleSyncDestination

The script class appears to already be in place.
```

5. Next, the Synchronization Server prompts if you want to configure any user-defined arguments needed by the server extension. Typically, you would define connection arguments, such as hostname, port, bindDN, or bindPassword if the destination calls for these parameters. The configuration parameters that are allowed are defined by the extension itself and the values are stored in the server configuration. These properties can be modified using the `dsconfig` tool and the web console. If there are user-defined arguments, enter "yes". Otherwise press **Enter** to accept the default (no) and continue. For this example, enter "yes" to configure the arguments for the ExampleSyncDestination.groovy script.

```
Do you need to configure any arguments for
com.unboundid.sdk.examples.groovy.ExampleSyncDestination? (yes / no) [no]: yes
```

6. Assuming you entered "yes" to configure any arguments, enter "n" to add a new argument. Then enter an extension argument in the form "name=value." For example, you can set the argument for the listener port, port=389. Repeat this step for any other arguments defined in your server extension.

7. Next, you will be prompted to configure the maximum number of before-and-after values for all changed attributes. Notification mode requires that the source change logs include all of the before-and-after values for changed attributes. Some entries, such as groups, might have attributes with hundreds or thousands of values, which could lead to excessively large change log entries, when all values are included in the changelog (the individual changes such as a user that is added or removed from a group are always included in the changelog entry). The cap is provided as a safeguard to avoid this problem; however, it is recommended that you set it to something well above the maximum number of values that any synchronized attribute will have. If this cap is exceeded, the Synchronization Server will issue an alert. For this example, we accept the default value of 200.

```
Enter a value for the max changelog before/after values,
or -1 for no limit [200]:
```

8. Next, you will be prompted to configure any key attributes in the change log that should always be included in every notification. These attributes can be used to find the destination entry corresponding to the source entry and will be present whether or not the attributes changed. In a later step, you will configure one or more Sync Classes, and any attributes you plan to use in a Sync Class include-filter should also be configured as key attributes. For this example, press **Enter** to add a key attribute, and then enter "`n`" to add a new key attribute. Then, enter "`uid`" as an example. Repeat this step to enter more entry key attributes.

```
Enter an attribute name: uid
```

9. Next, you will be prompted if you want the changes to be processed by the Sync engine strictly in sequential order, which will cause the worker threads to be reduced to 1. In both standard and notification modes, the Sync Pipe processes the changes concurrently with multiple threads, resulting in higher overall throughput, but make certain assurances about changes to the same entry being processed sequentially. If changes must be applied strictly in order, then the number of Sync Pipe worker threads will be reduced to 1. Note that this will limit the maximum throughput of the Sync Pipe, especially with a slow or remote destination endpoint.

Step 2. Configuring the Sync Pipe and Sync Classes

From this point on, the configuration steps follows the same process as a standard synchronization mode sync pipe. See *About the Sync User Account* for more information.

To Configure the Sync Pipe

1. Continuing from the previous session, enter a name for the Sync Pipe. Make sure the name is descriptive to identify it if you have more than one sync pipe configured. For example, enter "`ds-to-syncdest`".

2. Next, on the Sync Pipe Sync Class Definitions menu, you will be prompted if you would like to define one or more Sync Classes. Type `yes`.

To Configure the Sync Class

1. Next, enter a name for the Sync Class. Make sure the name is descriptive to identify the sync class.

2. At this stage, if you plan to restrict entries to specific subtrees, then enter one or more base DNs. For this example, press **Enter** to accept the default (no).

3. Next, you will be prompted to set an LDAP search filter. For this example, type `yes` to set up a filter and enter the filter `"(uid=*)"`. Press **Enter** again to continue. This property sets the LDAP filters and returns all entries that match the search criteria to be included in the Sync Class. In this example, we want to specify that any entry with an uid attribute be included in the Sync Class, regardless if there is a change or not to it.

4. Continuing from the previous example, on the Sync Class menu, you will be prompted if you want to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization. Press **Enter** to accept the default (all). You can adjust these mappings in a later section.

5. Next, specify the operations that will be synchronized for the Sync Class. For this example, press **Enter** to accept the default (1, 2, 3) for creates, deletes, modifies.

6. Review the configuration, and then press **Enter** to write the configuration to the Synchronization Server. If you want to change any property, you can go back to the particular menu, or make the adjustments later using the dsconfig tool. If you decide to write the configuration to the Synchronization Server, press **Enter**, and then enter the connection properties for your Synchronization Server (bindDN, bindPassword).

7. The `create-sync-pipe-config` tool outputs the final processing messages. If you have to make any manual changes to the external servers, it will present them. At this stage, you have successfully completed configuring your sync class.

Step 3. Configure Attribute and DN Mappings

At this point, you can set up your attribute and DN mappings for your sync pipe. The notifications procedure is identical to that of any standard mode implementation. For more information, see *Configuring Attribute Maps* and *Configuring DN Maps*.

Step 4. Configure Advanced Properties

Next, configure any advanced properties for your Sync Pipe in notification mode deployment using the `dsconfig` tool and accessing the Sync Class.

Step 5. Set the Startpoint in the Change Log

The `realtime-sync set-startpoint` command sets the starting point in the change log to tell the Synchronization Server where to start when the Sync Pipe is started. This command provides a way to avoid syncing all of the changes that have happened in the past.

To Set the Startpoint

- Run the `realtime-sync set-startpoint` command to an appropriate place in the change log. For example, the following command rewinds the startpoint at 15 minutes before the current time period.

```
$ realtime-sync set-startpoint --startpoint-rewind 15m \
--pipe-name "ds-to-syncdest" --bindPassword password --no-prompt
```

Step 6. Start the Sync Pipe

At this stage, we have configured everything necessary for the ds-to-syncdest Sync Pipe. We only need to start it. Generally, it is preferable to use the `realtime-sync` tool to start and stop the Sync Pipes as well as start and stop the Sync configuration globally.

To Start the Sync Pipe

- Run the `realtime-sync` tool to start Sync Pipe.

```
$ bin/realtime-sync start --pipe-name ds-to-syncdest
```

Step 7. Debugging the Configuration

Typically, you will need to debug any problems after you run the `prepare-endpoint-server` command. There are a number of logging and tools options available when debugging the configuration as follows:

Check the Status

- Run the `status` tool to verify the source-side connectivity and processing. You should check if the servers are connected and that changes are being detected. You can enter your bindDN password and have the system use your bind DN and port as defaults. For a description of each status parameter shown, see *Running the Status Tool*.

```
$ status --bindPassword password
```

- You can also restrict the status output to just list a single sync pipe using the `--pipe-name` option.

```
$ status --bindPassword password --pipe-name ds-to-syncdest
```

Check the Logs

- Increase the detail in the Sync log by changing the Sync Log Publisher handler's `logged-message-type` property to include: `change-applied-detailed`, `change-detected-detailed`, and `entry-mapping-details`. However, these properties should be disabled for production deployments as they could affect performance. The Synchronization Server records errors in the sync log if it detects change log entries that are missing information that are needed to perform a notification.

```
$ dsconfig --no-prompt set-log-publisher-prop \
--publisher-name "File-Based Sync Logger" \
--set logged-message-type:change-applied-detailed \
--set logged-message-type:change-detected-detailed \
--set logged-message-type:change-failed-detailed \
--set logged-message-type:dropped-op-type-not-synchronized \
--set logged-message-type:dropped-out-of-scope \
--set logged-message-type:entry-mapping-details \
--set logged-message-type:no-change-needed
```

- Enable the debug logger (disabled by default). You should disable the logger when no longer needed as it can impact performance.

```
# Enable the Debug Logger
dsconfig --no-prompt set-log-publisher-prop \
--publisher-name "File-Based Debug Logger" --set enabled:true

# Set the Debug Target and Verbosity Level
dsconfig --no-prompt create-debug-target \
--publisher-name "File-Based Debug Logger" \
--target-name com.unboundid.directory.sync.jdbc
--set debug-level:verbose

# When finished with debugging, disable the logger
dsconfig --no-prompt set-log-publisher-prop \
--publisher-name "File-Based Debug Logger" \
--set enabled:false
```

- If your connections are working and the `realtime-sync` operation is working but you are seeing sync errors, check the sync log. The problems could be in your attribute or DN maps.

Check the Alerts

- **Set an Alert for a Backlog of Changes**. If destination processing slows down, the sync worker threads can get backed up. You can set a property on the Sync Source Change Log

configuration to send an alert if a specified number of changes have been backed up. Once this number or threshold value has been exceeded, the Sync Source will send an alert.

```
$ dsconfig --no-prompt set-sync-source-prop \
--source-name "UnboundID Directory Server Source" \
--set sync-backlog-alert-threshold:5000
```

When to Restart the Sync Pipe

- Make sure to re-start the Sync Pipes after modifying a script implementation. Any Synchronization Server configuration change automatically re-starts the Sync Pipe. Script implementation changes require a manual Sync Pipe restart but no server restart. Java implementations require a server restart.

```
$ bin/realtime-sync stop
$ bin/realtime-sync start
```

Access Control Filtering on the Sync Pipe

As of version 3.2, the Synchronization Server provides an advanced Sync Pipe configuration property, `filter-changes-by-user`, that performs access control filtering on the target entry of a changelog entry for a specific user.

Administrators can configure a Sync Pipe in notification mode that performs access control filtering on the changelog data as it comes back from the source directory server. In this case, since the changelog entry contains data from the target entry, the access controls filter out attributes that the user does not have the privileges to see before it is returned. For example, values in the changes, `ds-changelog-before-values`, `ds-changelog-after-values`, `ds-changelog-entry-key-attr-values`, and `deletedEntryAttrs` attributes after filtered out through access control instructions.

This property is only available for Notification mode and can be configured using the `create-sync-pipe-config` or the `dsconfig` tool.

The source server must be the UnboundID Directory Server or Alcatel-Lucent 8661 Directory Server (version 3.2 or later), or an UnboundID Directory Proxy Server (version 3.2 or later) or Alcatel-Lucent 8661 Directory Proxy Server (version 3.2 or later) that points to an UnboundID Directory Server or Alcatel-Lucent 8661 Directory Server (version 3.2 or later).

Important Points about Access Control Filtering

Note the following points about access control filtering:

- The Directory Server will not return the changelog entry if the user is not allowed to see the target entry itself.
- The Directory Server strips out any attributes (for example, values in the changes, `ds-changelog-before-values`, `ds-changelog-after-values`, `ds-changelog-entry-key-attr-values`, and `deletedEntryAttrs` attributes) that the user is not allowed to see.

- If no changes are left in the entry, then no changelog entry will be returned.
- If only some attributes are stripped out, then the changelog entry will still be returned.
- Access control filtering on a specific attribute value is not supported. You will either get all attribute values or none.
- If a sensitive attribute policy is used to filter attributes when a client normally accesses the directory server, this sensitive attribute policy will not be taken into consideration during notifications since the Sync User is always connecting using the same method. You should configure your access controls in way to filter out these attributes not based on the type of connection made to the server but rather based on who is accessing the data. This way the `filter-changes-by-user` property will be able to evaluate if that person should have access to these attributes or not in the changelog entry for notifications.

To Configure the Sync Pipe to Filter Changes by Access Control Instructions

1. Set the `filter-changes-by-user` property to filter changes based on access controls for a specific user.

```
$ bin/dsconfig set-sync-pipe-prop --pipe-name "Notifications Sync Pipe" \
--set "filter-changes-by-user:uid=admin,dc=example,dc=com"
```

2. On the source Directory Server, set the report-excluded-changelog-attributes property to include the names of users that have been removed through access control filtering. This will allow the Synchronization Server to warn about attributes that were supposed to be synchronized but were filtered out. This step is recommended but not required.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
--set "report-excluded-changelog-attributes:attribute-names"
```

Note: The Synchronization Server only uses the attribute-names setting for the Directory Server's `report-excluded-changelog-attributes` property. It does not use the `attribute-counts` setting for the property.

Contact Your Support Provider

If you require technical support, your authorized support provider requests that you run the `bin/collect-support-data` command so that they can locate the source of any problems. The command generates a zip file that you can send to provier.

```
$ bin/collect-support-data --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

Chapter 8 Configuring Synchronization with SCIM

The UnboundID Synchronization Server provides data synchronization between directory servers or proxy servers and Simple Cloud Identity Management (SCIM) applications over HTTP. You can synchronize with custom SCIM applications or with the UnboundID Directory and Directory Proxy Server configured as SCIM servers using the SCIM extension.

Before setting up the Synchronization Server, review the section “Configuration Model” on page 17 to understand the important components of the Synchronization Server.

This chapter presents the following topics:

Topics:

- *About Synchronizing with a SCIM Sync Destination*
- *Configuring Synchronization with SCIM*
- *Mapping LDAP Schema to SCIM Resource Schema*

About Synchronizing with a SCIM Sync Destination

You can configure the Synchronization Server to synchronize with SCIM service providers. The Simple Cloud Identity Management (SCIM) protocol is designed to make managing user identity in cloud-based applications and services easier. SCIM allows you to provision identities, groups, and passwords to, from, and between clouds.

Note: You can configure the UnboundID Directory and Directory Proxy Servers to be SCIM servers using the SCIM Extension bundle. For more information about configuring the SCIM Extension for use with the UnboundID Directory and Directory Proxy Server, see the UnboundID SCIM Extension User's Guide.

The Synchronization Server is LDAP-centric and operates on LDAP attributes. The SCIM sync destination server component acts as a translation layer between a SCIM service provider's schema and an LDAP representation of the entries.

Note: While the Synchronization Server is LDAP-centric and typically at least one endpoint is an LDAP Directory Server, this is not a strict requirement. For example, you could set up a JDBC to SCIM sync pipe.

The Synchronization Server contains sync classes that define how source and destination entries are correlated. The SCIM sync destination contains its own mapping file, `scim-resources.xml`, that maps LDAP schema to SCIM.

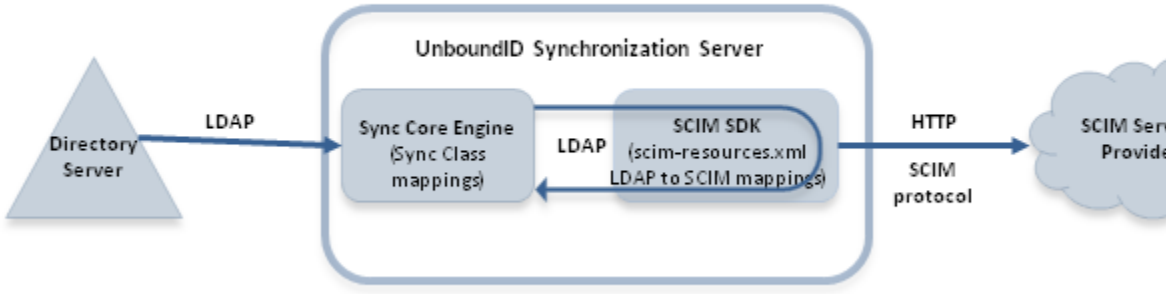


Figure 27: Synchronizing with a SCIM Sync Destination

The SCIM destination supports high availability and failover and SSL communication. As for other types of endpoint, you can configure SCIM sync destinations using the `create-sync-pipe-config` tool.

Note: The Synchronization Server can only use SCIM as a Sync Destination. Note There is no mechanism in the SCIM protocol for detecting changes, so it cannot be used as a Sync Source.

Overview of SCIM Destination Configuration Objects

The `SCIMSyncDestination` object defines a SCIM service provider sync pipe destination that is accessible over HTTP via the SCIM protocol. It is configured with the following properties:

- **server**. Specifies the names of the SCIM External Servers that are used as the destination of synchronization.
- **resource-mapping-file**. Specifies the path to the scim-resources.xml file, a configuration file that defines the SCIM schema and maps it to the LDAP schema. Out of the box, this file is located in `<server root>/config/scim-resources.xml`. This file can be customized to define and expose deployment-specific resources. For information about this file and how to map resources to and from the LDAP entries, refer to the SCIM SDK and Reference Implementation documentation at http://www.unboundid.com/labs/projects/simple-cloud-identity-management-scim/docs/scim-sdk-docs.
- **rename-policy**. Specifies how to handle the rename of a SCIM resource.

The SCIM Sync Destination object is based on the open source UnboundID SCIM SDK. Before configuring a SCIM destination, you may want to familiarize yourself with the following documents. They will help you understand and make efficient use of SCIM with the Synchronization Sever.

- SCIM Core Schema: http://www.simplecloud.info/specs/draft-scim-core-schema-02.html
- SCIM REST API: http://www.simplecloud.info/specs/draft-scim-rest-api-01.html

Tips for Syncing to a SCIM Destination

When configuring an LDAP to SCIM Sync Pipe, you should be aware of the following:

- **Use scim-resources.xml for Attribute and DN Mappings**. We recommend that you avoid creating attribute and DN mappings for the Sync class associated with the directory-to-SCIM sync pipe. When working with SCIM sync destinations, there are two layers of mapping, once at the Sync Class level and again at the SCIM sync destination level in the scim-resources.xml file. To reduce complexity, do all the mappings that you can in the `scim-resources.xml` file.
- **Avoid Groups Unless the SCIM ID is DN Based**. Group synchronization is supported if the SCIM ID is based on the DN. If the SCIM ID is not the DN itself, it must be one of the components of the RDN, meaning that the DNs of group members must contain the necessary attribute. If a SCIM service provider uses `entryUUID` as the SCIM ID, then the Synchronization server creates or modifies the group entry in SCIM by looking up the `entryUUID` for each group member, which is not currently supported.

- **SCIM Modifies Entries Using PUT**. The SCIM sync destination modifies entries using the full HTTP PUT method. For every modify, SCIM replaces the entire resource with the updated resource. For information about the implications of this on password updates, refer to “Password Considerations with SCIM”.

Renaming a SCIM Resource

The SCIM protocol does not support changes that require the SCIM resource to be renamed, such as a MODDN operation. Instead, when a change is detected to an attribute value that is used as part of the SCIM ID attribute, the Synchronization Server handles it in one of the following ways:

- Deletes the specified SCIM resource and then adds the new resource with the new SCIM ID.
- Adds the new resource with the new SCIM ID and then deletes the old resource.
- • Skips the rename portion of the change. If renames are expected on the source endpoint, a careful set of destination-correlation attributes should be chosen so that the destination can still be found after it is renamed on the source.

You can configure this behavior by setting the `rename-policy` property of the SCIM Sync Destination.

Password Considerations with SCIM

Because the SCIM sync destination modifies entries using a full PUT method, special considerations need to be made for password attributes. The UnboundID SCIM Server allows password attributes to be omitted from a change when they have not been modified by an operation. This prevents passwords from inadvertently being overwritten during the PUT operation, which does not include the password attribute. Ideally, other SCIM service providers will not wipe a password because a PUT request does not contain it. Check with your vendor to confirm this behavior before starting your SCIM sync pipe.

Configuring Synchronization with SCIM

You can configure synchronization with SCIM using the create-sync-pipe-config utility or using the `dsconfig` command-line tool. If you are configuring from scratch, we recommend using the `create-sync-pipe-config` tool as it will lead you through the steps necessary to define each component.

If you configure synchronization between an LDAP server and a SCIM service provider from scratch, you need to take the following steps:

- **Set up External Servers**. Configure one external server for every physical endpoint.
- **Configure the Sync Source server**. Designate the external servers that correspond to the source server.

- **Configure the Sync Destination server**. Designate the external servers that correspond to the SCIM sync destination.
- **Configure the Sync Pipe**. Configure tyour LDAP to SCIM sync pipe.
- **Configure the Sync Classes**. Each sync class represents a type of entry that needs to be synchronized. When specifying a sync class for synchronization with a SCIM service provider, you want to avoid including attribute and DN mappings, but instead use it to specify operations that you want to synchronize and which correlation attributes to use.
- **Set the Evaluation Order for your Sync Classes**. Each sync class must be assigned an evaluation order to determine the processing precedence for each class.
- **Set Up Communication for each External Server**. Run `prepare-endpoint-server` once for every LDAP external server that is part of the sync source.
- **Start Sync**. Use `realtime-sync` to set the startpoint and then start the sync pipe.

Configuring the External Servers

Before you begin, you first need to set up an external server for each host in your deployment. This entails registering the directory server as the Sync Source server and the SCIM server as the Sync Destination server.

To Configure the External Servers

1. Configure the UnboundID Directory Server as an external server, which will later be configured as a Sync Source. On the Synchronization Server, run the following `dsconfig` command:

```
$ bin/dsconfig create-external-server \
  --server-name source-ds \
  --type unboundid-ds \
  --set server-host-name:ds1.example.com \
  --set server-port:636 \
  --set "bind-dn:cn=Directory Manager" \
  --set password:secret \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:JKS
```

2. Configure the SCIM Server as an external server, which will later be configured as a Sync Destination. The `scim-service-url` property specifies the location of the SCIM sync destination, which is the complete URL used to access the SCIM service provider. The `user-name` property provides the account used to connect to the SCIM service provider. It is used in conjunction with the chosen authentication method. By default, the value is set to `cn=Sync User,cn=Root DNs,cn=config`. Note that for other SCIM service providers, the user name might not be in DN format.

```
$ bin/dsconfig create-external-server \
  --server-name scim \
  --type scim \
  --set scim-service-url:https://scim1.example.com:8443 \
  --set "user-name:cn=Sync User,cn=Root DNs,cn=config" \
  --set password:secret \
  --set connection-security:ssl \
  --set hostname-verification-method:strict \
```

```
  --set trust-manager-provider:JKS
```

Configuring the Directory Server Sync Source

At this stage, you need to configure the Sync Source for your synchronization network. You can configure more than one external server to act as the sync source for failover purposes. If the source is an UnboundID Directory Server, you must also configure the following items:

- **Enable Changelog Password Encryption Plug-in**. You need to enable the change log password encryption plugin on any directory server that will receive password modifications. This plugin intercepts password modifications, encrypts the password and adds an encrypted attribute to the change log entry.
- **Synchronizing Deletes.**You need to configure the `changelog-deleted-entry-include-attribute` property on the changelog backend, so that the Synchronization Server can properly record which attributes were removed during a DELETE operation.

To Configure the Directory Server Sync Source

1. Run `dsconfig` to configure the external server as the Sync Source. Based on the previous example where we configured the UnboundID Directory Server as `source-ds`, run the following command:

```
$ bin/dsconfig create-sync-source --source-name source \
  --type unboundid \
  --set base-dn:dc=example,dc=com \
  --set server:source-ds \
  --set use-changelog-batch-request:true
```

2. Enable the change log password encryption plugin on any directory server that will receive password modifications. You can copy and paste the encryption key from the output, if displayed, or access it from the `<server-root>/bin/sync-pipe-cfg.txt` file, if you used the `create-sync-pipe-config` tool to set up your sync pipe.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Changelog Password Encryption"
  --set enabled:true\
  --set changelog-password-encryption-key:ej5u9e39pqo68
```

3. Enable the change log password encryption plug-in on any directory server that will receive password modifications. This plugin intercepts password modifications, encrypts the password and adds an encrypted attribute to the change log entry. You can copy and paste the encryption key from the output, if displayed, or access it from the `<server-root>/bin/sync-pipe-cfg.txt` file, if you used the `create-sync-pipe-config` tool to set up your sync pipe.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Changelog Password Encryption" \
  --set enabled:true\
  --set changelog-password-encryption-key:ej5u9e39pqo68
```

4. Next, on the sync server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pq-68
```

5. Finally, configure the `changelog-deleted-entry-include-attribute` property on the changelog backend.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

Configuring the SCIM Sync Destination

The SCIM sync destination synchronizes data with a SCIM service prpovider.

To Configure the SCIM Sync Destination

- Run the `dsconfig` command to configure the SCIM external serer as the Sync Destination.

```
$ bin/dsconfig create-sync-destination \
  --destination-name scim \
  --type scim \
  --set server:scim
```

Configuring the Sync Pipe, Sync Classes, and Evaluation Order

This section describes how to configure a sync pipe for LDAP to SCIM synchronization, how to create sync classes for the sync pipe, and how to set the evaluation order index for the sync classes.

Note: The Synchronization mode must be set to Standard. You cannot currently use Notification Mode with SCIM.

To Configure the SCIM Sync Pipe

Once you have configured the source and destination endpoints, you can configure the sync pipe for your LDAP to SCIM synchronization.

- Run `dsconfig` command to configure the LDAP-to-SCIM Sync Pipe.

```
$ bin/dsconfig create-sync-pipe \
  --pipe-name ldap-to-scim \
  --set sync-source:source \
  --set sync-destination:scim
```

To Configure the SCIM Sync Classes

Next, we create three sync classes. The first sync class is used to match user entries in the Sync Source. The second class is used to match group entries. The third class is used to a DEFAULT class that is used to match all other entries.

1. Run the `dsconfig` command to create the Sync Class. In the following command, you set the Sync Pipe Name and Sync Class name.

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
  --class-name user
```

2. Next, use `dsconfig` to set the base DN and filter for the Sync Class that was created in the previous step. The `include-base-dn` property specifies a the base DN in the source, which is `ou=people,dc=example,dc=com`. So, this sync class is invoked only for changes at the `ou=people` level. The `include-filter` property specifies an LDAP filter that tells the synchronization server to include `inetOrgPerson` entries as user entries.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --add include-base-dn:ou=people,dc=example,dc=com \
  --add "include-filter:(objectClass=inetOrgPerson)"
```

3. Next, create the sync class, which is used to match group entries.

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
  --class-name group
```

4. For the second Sync Class, set the base DN and the filters to match the group entries.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name group \
  --add include-base-dn:ou=groups,dc=example,dc=com \
  --add  "include-filter:(|(objectClass=groupOfEntries)\
    (objectClass=groupOfNames)(objectClass=groupOfUniqueNames)\
    (objectClass=groupOfURLs))"
```

5. For the third Sync Class, create a DEFAULT Sync Class that is used to match all other entries. Because we do not want to synchronize changes that come from anything but user and group entries, we set `synchronize-creates`, `synchronize-modifies`, and `synchronize-delete` to `false`. Alternatively, you can omit this class, as entries that do not match a sync class are not synchronized.

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
  --class-name DEFAULT \
  --set evaluation-order-index:99999 \
  --set synchronize-creates:false \
  --set synchronize-modifies:false \
  --set synchronize-deletes:false
```

To Set the Evaluation Order Index

Once you have configured all of the sync classes needed by your sync pipe, you set the evaluation order index for each sync class. The sync pipe uses the evaluation order index to decide which sync class to process first. Classes with a lower number are evaluated first.

- Run `dsconfig` to set the evaluation order index for the Sync Class. Classes with a lower number are evaluated first. In this example, set the value to 100. The actual number depends on your particular deployement.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --set evaluation-order-index:100
```

Setting Up Communication with the Source Server(s)

Next, use the `prepare-endpoint-server` tool to set up communication between the synchronization server and the LDAP source servers. If user accounts do not exist, this tool creates the appropriate user account and its privileges for the Synchronization Server to use. Also, because the source is a Directory Server, this tool enables the change log.

Note: The `prepare-endpoint-server` tool can only be used on LDAP directory servers. For the SCIM Server, you must manually create a sync user entry.

To Set Up Communication with the Source Server(s)

Run the `prepare-endpoint-server` command to setup communication with the Synchronization Server and the source server(s). The tool will then prompt you for the bind DN and password to create the user account and enables the change log.

```
$ bin/prepare-endpoint-server \
  --hostname ds1.example.com \
  --port 636 \
  --useSSL \
  --trustAll \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword "password" \
  --baseDN "dc=example,dc=com" \
  --isSource
```

Starting the Sync Pipe

The `realtime-sync` tool sets a specific starting point for real-time synchronization, so that changes made before the current time are ignored, and schedules a stop or start at a future date.

To Start and Manage the SCIM Sync Pipe

1. Run the `realtime-sync` tool to set the startpoint for the Sync Source.

```
$ bin/realtime-sync set-startpoint \
  --end-of-changelog \
  --pipe-name ldap-to-scim
```

2. Once you are ready to start synchronization, run the following command:

```
$ bin/realtime-sync start \
  --pipe-name ldap-to-scim \
  --no-prompt
```

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.

Note: The `scim-resources.xml` file is the same as the `resources.xml` file shipped with the SCIM Extension for the UnboundID Directory Server.

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.

The default configuration maps the SCIM resource ID to the LDAP entry DN. However, the entry DN does not necessarily meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN (the DN can only be changed through the LDAP interface, not through the SCIM interface). The resource configuration allows the SCIM resource ID to be mapped to an LDAP attribute as an alternative to mapping to the LDAP entry DN. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, is a possible alternative mapping. However, configuring a mapping to an attribute, rather than the entry DN, may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the directory information tree. If using the UnboundID SCIM Extension bundle for the UnboundID Directory Server, this can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique resources.xml configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique resources.xml configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

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

A `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.

A **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 elements in sequence:

- **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 attributes:

- **tag**: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A **complexMultiValued** element contains the following XML elements in sequence:

- **subAttribute**: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- **canonicalValue**: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

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 attributes:

- **name**: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML elements in sequence:

- **subMapping**: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- **ldapAttribute**: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- **transform**: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in "*Mapping LDAP Schema to SCIM Resource Schema*".

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 Schema to SCIM Resource Schema".

About the <LDAPSearch> Element

A `LDAPSearch` element has the following XML attributes:

- **baseDN**: a required element specifying the LDAP search base DN to be used when querying for the SCIM resource.
- **filter**: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- **resourceIDMapping**: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN.

Note:

The LDAPSearch element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an `ID` attribute.

About the <resourceIDMapping> Element

A `resourceIDMapping` element has the following XML attributes:

- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- **createdBy**: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include scim-consumer, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or directory, meaning that a value is automatically provided by the Directory Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an LDAPSearch element that contains a resourceIDMapping element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

A `LDAPAdd` element has the following XML attributes:

- **DNTemplate**: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using {ldapAttr}, where ldapAttr is the name of an LDAP attribute.
- **fixedAttribute**: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

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 the following XML element:

- **fixedValue**: one or more elements specifying the fixed LDAP values.

Chapter 9 Managing Logging and Alerts

The Synchronization Server supports extensive logging features to track any aspect of your Synchronization topology. You can also set up administrative alert handlers to notify of any specific events.

This chapter presents the following information:

Topics:

- *Working with Logs*
- *Default Synchronization Server Logs*
- *Viewing the List of Log Publishers*
- *Sync Log Message Types*
- *Creating New Log Publishers*
- *Configuring Log Rotation*
- *Configuring Log Retention*
- *Working with Administrative Alert Handlers*
- *Configuring the SNMP Subagent Alert Handler*
- *Running the Status Tool*
- *Monitoring the Synchronization Server*
- *Monitoring Using SNMP*

Working with Logs

The UnboundID® Synchronization Server supports different types of log publishers that can be used to provide the monitoring information for sync, access, debug, and error messages that occur during normal server processing. The Synchronization Server provides a standard set of default log files as well as mechanisms to configure custom log publishers with their own log rotation and retention policies.

Types of Log Publishers

The UnboundID Synchronization Server provides a number of different types of log publishers that can be used to log processing information about the server. There are several primary types of loggers:

- **Sync loggers** provide information about synchronization actions that occur within the server. Specifically, the Sync Log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization. The log also shows the entries that were involved in the synchronization process.
- **Resync loggers** provide summaries or details of synchronized entries and any missing entries in the Sync Destination.
- **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.
- **Access loggers** provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the UnboundID Sync Management console.

By default, the following log publishers are enabled on the system:

➢ File-based sync logger
➢ File-based access logger
➢ File-based error logger

The UnboundID Synchronization Server also provides a File-based Audit Logger, which is a special type of access logger that can provide detailed information about changes processed within the server, and a File-based Debug Logger. Both are disabled by default.

Default Synchronization Server Logs

The Synchronization Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the `UnboundID-Sync/logs` directory. The following default log files are available as seen in the table below.

Table 19: Synchronization Server Logs

Log File	Description
access	File-based Access Log that records LDAP operations processed by the Synchronization Server. Access log records can be used to provide information about problems during operation processing and provide information about the time required to process each operation.
config-audit.log	Records information about changes made to the Synchronization Server configuration in a format that can be replayed using the dsconfig tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that are not errors but occur during server processing.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to server.out.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a Synchronization server with the setup command.
sync	File-based Sync Log that records synchronization operations processed by the server. Specifically, the log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization.
sync-pipe-cfg.txt	Records the configuration changes used with the `bin/create-sync-pipe-config` tool. The file is placed wherever the tool is run. Typically, this is in server-root or in the `bin` directory.
tools	Holds logs for long running utilities. Current and previous copies of the log are present in the directory.
update.log	Records messages that occur during a Synchronization Server upgrade.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Synchronization Server using the `dsconfig` tool.

Note: Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using `dsconfig` before they appear in the list of

log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

- Use `dsconfig` to view the log publishers.

```
$ bin/dsconfig list-log-publishers

Log Publisher                                : Type              : enabled
---------------------------------------------:-------------------:--------
Expensive Operations Access Logger           : file-based-access : false
Failed Operations Access Logger              : file-based-access : true
File-Based Access Logger                     : file-based-access : true
File-Based Audit Logger                      : file-based-audit  : false
File-Based Debug Logger                      : file-based-debug  : false
File-Based Error Logger                      : file-based-error  : true
Replication Repair Logger                    : file-based-error  : true
Successful Searches with No Entries Returned : file-based-access : false
```

Sync Log Message Types

The Synchronization Server logs certain types of log messages with the sync log. You can control which message types can be included or excluded from the logger, or added to in a custom log publisher.

Table 20: Sync Log Message Types

Message Type	Description
change-applied	Default summary message. Logged each time a change is applied successfully.
change-detected	Default summary message. Logged each time a change is detected.
change-failed-detailed	Default detail message. Logged when a change cannot be applied. It includes the reason for the failure and details about the change that can be used to manually repair the failure.
dropped-op-type-not-synchronized	Default summary message. Logged when a change is dropped because the operation type (for example, ADD) is not synchronized for the matching Sync Class.
dropped-out-of-scope	Default summary message. Logged when a change is dropped because it does not match any Sync Class.
no-change-needed	Default summary message. Logged each time a change is dropped because the modified source entry is already in-sync with the destination entry.
change-detected-detailed	Optional detail message. Logged each time a change is detected. It includes attribute values for added and modified entries. This level of information is often useful for diagnosing problems, but it causes log files to grow faster, which impacts performance
entry-mapping-details	Optional detail message. Logged each time a source entry (attributes and DN) are mapped to a desintation entry. This level of information is often useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-applied-detailed	Optional detail message. Logged each time a change is applied. It includes attribute values for added and modified entries. This level of information is often

Message Type	Description
	useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-failed	Optional summary message. Logged when a change cannot be applied. It includes the reason for the failure but not enough information to manually repair the failure.
intermediate-failure	Optional summary message. Logged each time an attempt to apply a change fails. Note that a subsequent retry of applying the change might succeed.

Creating New Log Publishers

The UnboundID® Synchronization Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. View the Log Publishers.

```
$ bin/dsconfig list-log-publishers
```

```
Log Publisher           : Type           : enabled
------------------------:----------------:--------
Disconnect Logger       : file-based-access : true
File-Based Access Logger   : file-based-access : true
File-Based Audit Logger    : file-based-access : false
File-Based Debug Logger    : file-based-debug  : false
File-Based Error Logger    : file-based-error  : true
```

```
Replication Repair Logger  : file-based-error  : true
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the UnboundID® Synchronization Server Configuration console main menu, select the option to configure the log publisher.
4. On the **Log Publisher Management** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the directory server root. For example, `logs/disconnect.log`.
9. Select the rotation policy you want to use for your log publisher.
10. Select the retention policy you want to use for your log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring Log Rotation

The Synchronization Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Synchronization Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Retention

The Synchronization Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Synchronization Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Synchronization Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your directory server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Working with Administrative Alert Handlers

The UnboundID® Synchronization Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems

during server startup or shutdown. The Synchronization Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Synchronization Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. UnboundID uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the 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 Synchronization Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Running the Status Tool

The Synchronization Server provides a command-line tool, status, that outputs the health of the Synchronization server. The `status` tool is a command-line utility that polls the current health

of the server and displays summary information about the number of operations processed in the network. The tool provides different component categories as shown below.

Table 21: Status Tool Sections

Status Section	Description

Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
	
	`--- Server Status ---`
	`Server Run Status:  Started 17/May/2010:15:26:47.000-0500`
	`Operational Status: Available`
	`Open Connections:   6`
	`Max Connections:    8`
	`Total Connections:  24`

Server Details	Displays the server details including host name, administrative users, install path, Sync Server version, and Java version.
	
	`--- Server Details ---`
	`Host Name:            sync1.example.com`
	`Administrative Users: cn=ADSync User`
	`Administrative Users: cn=Directory Manager`
	`Administrative Users: cn=IntraSync User`
	`Installation Path:    /UnboundID-Sync`
	`Server Version:       UnboundID Synchronization Server 3.2.1`
	`Java Version:         Sun/Oracle JDK 1.6.0_31`

Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
	
	`--- Connection Handlers ---`
	`Address:Port : Protocol : State`
	`-------------:----------:---------`
	`0.0.0.0:1689 : JMX      : Disabled`
	`0.0.0.0:636  : LDAPS    : Disabled`
	`0.0.0.0:7389 : LDAP     : Enabled`

Sync Topology	Displays information about the connected Sync topology and any standby sync server instances.
	
	`--- Sync Topology ---`
	`Host:Port                          : Status  : Priority`
	`Index`
	`-----------------------------------:---------:---------------`
	`sync1.example.com:7389 (this server) : Active  : 1`
	`sync2.example.com:8389               : Standby : 2`

Summary for Sync Pipe	Displays the health status for each sync pipe configured on the topology, including current status, percent busy, changes detected, operations completed, number of operations processing, number of operations waiting, source unretrieved changes, failed operation attempts, source changes count. The most important stats to view are the Source Unretrieved Changes and the Failed Op Attempts. ➢ **Started**. Indicates whether the Sync Pipe has started or not. ➢ **Current Ops Per Second**. Indicates the current throughput rate in operations per second. ➢ **Percent Busy**. Indicates the number of sync operations currently in flight divided by the number of worker threads. ➢ **Changes Detected**. Indicates the total number of changes detected. ➢ **Ops Completed Total**. Indicates the toal number of changes detected and completed.

<table>
<tr><th>Status Section</th><th>Description</th></tr>
<tr><td></td><td>
➢ Num Ops In Flight. Indicates the number of operations that are in flight.
➢ Num Ops In Queue. Indicates the number of operations that are on the input queue waiting to be synchronized.
➢ Source Unretrieved Changes. Indicates how many outstanding changes are still in the source changelog that have not yet been retrieved by the Sync Server. If this is greater than zero, it indicates a sync backlog, because the internal sync queue is already too full to bring in these changes.
➢ Failed Op Attempts. Indicates the number of failed operation attempts.
➢ Poll For Source Changes Count. Indicates the number of times that the source has been polled for changes.

<pre>
     --- Summary for 'UBID1 to UBID2' Sync Pipe ---
Summary Stat   : Count
------------------------------:-------
Started   : true
Current Ops Per Second   : 1882
Percent Busy   : 0
Changes Detected   : 27299
Ops Completed Total   : 26746
Num Ops In Flight   : 0
Num Ops In Queue   : 0
Source Unretrieved Changes   : 10218
Failed Op Attempts   : 5
Poll For Source Changes Count : 480
</pre>
</td></tr>
<tr><td>Operations Completed for the Sync Pipe</td><td>
Displays the completed operation statistics for the sync pipe, including the number of successful operations, out of scope, operation type not synced, no change needed, entry already exists, no match found, multiple matches found, failed during mapping, failed at resource, unexpected exception, total operations.

➢ Success. Indicates the total number of changes that completed successfully.
➢ Out Of Scope. Indicates the total number of changes that made it into the Sync Pipe but were dropped because they did not match the criteria in a Sync Class.
➢ Op Type Not Synced. Indicates the total number of changes that completed because the operation type (e.g. create) is not synchronized.
➢ No Change Needed. Indicates the total number of changes that completed because no change was needed.
➢ Entry Already Exists. Indicates the total number of changes that completed unsuccessfully because the entry already existed for a create operation.
➢ No Match Found. Indicates the total number of changes that completed unsuccessfully because no match for an operation (e.g. a modify) was found.
➢ Multiple Matches Found. Indicates the total number of changes that completed unsuccessfully because multiple matches for a source entry were found at the destination.
➢ Failed During Mapping. Indicates the total number of changes that completed unsuccessfully because there was a failure during attribute or DN mapping.
➢ Failed At Resource. Indicates the total number of changes that completed unsuccessfully because they failed at the source.
➢ Unexpected Exception. Indicates the total number of changes that completed unsuccessfully because there was an unexpected exception during processing (e.g. an NPE).
➢ Total. Indicates the total number of operations completed.

<pre>
    --- Ops Completed for 'UBID1 to UBID2' Sync Pipe ---
Op Result   : Count
-----------------------:-------
Success   : 4559
Out Of Scope   : 0
Op Type Not Synced   : 0
No Change Needed   : 22181
</pre>
</td></tr>
</table>

Status Section	Description
	Entry Already Exists : 0 No Match Found : 0 Multiple Matches Found : 0 Failed During Mapping : 0 Failed At Resource : 0 Unexpected Exception : 0 Total : 26746
Sync Pipe Source Stats	Displays the source statistics for the external server, including the current connection status, successful connect attempts, failed connect attempts, forced disconnects, unretrieved changed, failed to decode changelog entry. ➢ **Is Connected**. Indicates whether the Sync Source is connected or not. ➢ **Connected Server**. Indicates the hostname and port number of the connected server. ➢ **Successful Connect Attempts**. Indicates the number of successful connection attempts. ➢ **Failed Connect Attempts**. Indicates the number of failed connection attempts. ➢ **Forced Disconnects**. Indicates the number of forced disconnects. ➢ **Root DSE Polls**. Indicates the number of polling attempts of the root DSE. ➢ **Unretrieved Changes**. Indicates the number of unretrieved changes. ➢ **Entries Fetched**. Indicates the number of entries fetched from the source. ➢ **Failed To Decode Changelog Entry**. Indicates the operations that failed to decode changelog entries. ➢ **Ops Excluded By Modifiers Name**. Indicates the number of operations excluded by modifier's name. ➢ **Num Backtrack Batches Retrieved**. Indicates the number of backtrack batches retrieved. --- Source Stats for 'UBID1 to UBID2' Sync Pipe --- Source Stat : Value ----------:---------- Is Connected : true Connected Server : ldap:// sync1.example.com:1389 Successful Connect Attempts : 1 Failed Connect Attempts : 8 Forced Disconnects : 1 Root DSE Polls : 366 Unretrieved Changes : 10218 Entries Fetched : 26740 Failed To Decode Changelog Entry : 0 Ops Excluded By Modifiers Name : 0 Num Backtrack Batches Retrieved : 0
Sync Pipe Destination Stats	Displays the destination statistics for the external server, including the current connection status, successful connect attempts, failed connect attempts, forced disconnects, unretrieved changed, failed to decode changelog entry. ➢ Is Connected. Indicates whether the Sync Source is connected or not. ➢ Connected Server. Indicates the connection URL of the connected server. ➢ Successful Connect Attempts. Indicates the number of successful connection attempts. ➢ Failed Connect Attempts. Indicates the number of failed connection attempts. ➢ Forced Disconnects. Indicates the number of forced disconnects. ➢ Entries Fetched. Indicates the number of entries fetched. ➢ Entries Created. Indicates the number of entries created. ➢ Entries Modified. Indicates the number of entries modified. ➢ Entries Deleted. Indicates the number of entries deleted.

Status Section	Description

	`--- Destination Stats for 'UBID1 to UBID2' Sync Pipe ---`
	`Source Stat : Value`
	`----------:----------`
	`Is Connected : true`
	`Connected Server : ldap://sync1.example.com:3389`
	`Successful Connect Attempts : 1`
	`Failed Connect Attempts : 0`
	`Forced Disconnects : 0`
	`Entries Fetched : 26740`
	`Entries Created : 0`
	`Entries Modified : 4559`
	`Entries Deleted : 0`

Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48 hour period. You can limit the number of displayed alerts using the `--maxAlerts` option. For example, `status --maxAlerts 0` suppresses any displayed alerts. `--- Administrative Alerts ---` `Severity : Time : Message` `----------:----------:----------` `Informational : 17/May/2010 15:28:07 -0500 : A configuration change has been made in the Synchronization Server: [17/May/2010:15:28:07:-0500] conn=1 op=7 dn='cn=Directory Manager,cn=Root DNs,cn=config' authtype=[Simple] from=10.2.1.232 to=101.6.1.232 command='dsconfig createexternal-server --server-name sync1.example.com:1389 --type unboundid --set server-hostname: sync1.example.com --set server-port:1389 --set 'bind-dn:cn=Sync User,cn=Root DNs,cn=config' --set 'password:AAB8sw4PRiOfLXNI4cu+5Saa''` `Informational : 17/May/2010 15:26:47 -0500 : The Synchronization Server has started successfully` `Shown are info messages, warnings and errors from the past 48 hours`

To Run the Status Tool

- Go to the installation directory. Run the `status` command on the command line.

```
$ bin/status --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
```

To Search for a Specific Status Monitor

- You can use the ldapsearch utility to directly search for a specific monitoring statistic. For example, run `ldapsearch` to find the current throughput of a Sync Pipe.

```
$ sync1/bin/ldapsearch --hostname sync.example.com --port 30636 \
--baseDN "cn=Sync Pipe Monitor: ds-to-dsml,cn=monitor" \
--searchScope base "(objectClass=*)" "current-ops-per-second"

Arguments from tool properties file:  --useSSL true --bindDN cn=Directory Manager
--bindPassword ***** --trustAll true

dn: cn=Sync Pipe Monitor: ds-to-dsml,cn=monitor
current-ops-per-second: 1001
```

Monitoring the Synchronization Server

The UnboundID Synchronization Server exposes its monitoring information under the cn=monitor entry for easy access to its information. Administrators can use various means to monitor the server's information including the Synchronization Management Console, JConsole, LDAP command-line tools, and through SNMP.

Table 22: Synchronization Server Monitoring Component

Component	Description
Active Operations	Provides information about the operations currently being processed by the Synchronization Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backend	Provides general information about the state of a Synchronization Server backend, including the backend ID, base DN(s), entry counts, entry count for the cn=admin data, writability mode, and whether it is a private backend. The following backend monitors are provided: ➢ adminRoot ➢ ads-truststore ➢ alerts ➢ backup ➢ config ➢ monitor ➢ schema ➢ tasks ➢ userRoot
Berkeley DB JE Environment	Provides information about the state of the Oracle Berkeley DB Java Edition database used by the Synchronization Server backend. Most of the statistics are obtained from the Berkeley DB JE and are not under the control of the Synchronization Server.
Client Connections	Provides information about all client connections to the Synchronization Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Disk Space Usage	Provides information about the disk space available to various components of the Synchronization Server.
Connection Handler	Provides information about the available connection handlers on the Synchronization Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
General	Provides general information about the state of the Synchronization Server, including product name, vendor name, server version, etc.
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 Synchronization Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Synchronization Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Synchronization Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, number of busy worker threads, etc.

Monitoring Using SNMP

The UnboundID® Synchronization Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Synchronization Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Synchronization Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The UnboundID® Synchronization Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Synchronization Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Synchronization Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

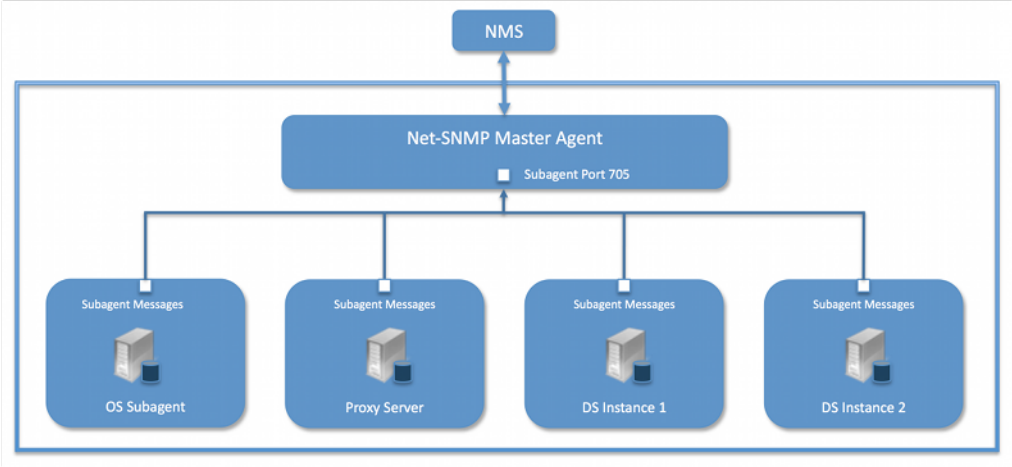


Figure 28: Example SNMP Deployment

One important note is that the UnboundID® Synchronization Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module, such as the System Management Agent (SMA) on Solaris or OpenSolaris. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Synchronization Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Synchronization Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Synchronization Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30

bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin.

```
$ bin/dsconfig set-plugin-prop --plugin-name "SNMP Subagent" \
  --set enabled:true --set agentx-address:localhost \
  --set agentx-port:705 --set session-timeout:5s \
  --set connect-retry-max-wait:10s
```

2. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

3. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

4. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

5. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

6. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

7. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

8. Start the `snmpd` daemon and after a few seconds you should see the following message in the Synchronization Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

9. Set up a trap client to see the alerts that are generated by the Synchronization Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

10. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

11. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (`-f`), do not read any configuration files (`-C`) except the one specified with the `-c` option, print to standard output (`-Lo`), and then specify that debugging output should be turned on for the User-based Security Module (`-Dusm`). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

12. Run the Net-SNMP client tools to test the feature. The following options are required: `-v <SNMP version>`, `-u <user name>`, `-A <user password>`, `-l <security level>`, `-n <context name (instance name)>` . The `-m` all option loads all MIBs in the default MIB directory in `/usr/share/snmp/mibs` so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost systemStatus
```

13. If you want alerts sent from the SNMP Subagent through the Net-SNMP master agent and onwards, you must enable the SNMP Subagent Alert Handler. The SNMP Alert Handler is used in deployments that do not enable the Subagent.

```
$ bin/dsconfig --no-prompt set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Configuring SNMP on AIX

Native AIX SNMP implementations do not support AgentX sub-agents, which is a requirement for UnboundID® Synchronization Servers. To implement SNMP on AIX platforms, any freely-available net-snmp package must be installed.

Special care must be made to ensure that you are using the net-snmp binary packages and not the native snmp implementation. Third-party net-snmp binary packages typically install under `/opt/freeware` and have the following differences:

```
Native Daemon: /usr/sbin/snmpd
Native Configuration File: /etc/snmpd.conf, /etc/snmpdv3.conf
Native Daemon Start and Stop: startsrc -s snmpd, stopsrc -s snmpd

net-snmp Daemon: /opt/freeware/sbin/snmpd
net-snmp Configuration File: /opt/freeware/etc/snmp/snmpd.conf
net-snmp start and stop: /etc/rc.d/init.d/snmpd start|stop
```

When configuring an SNMP implementation on AIX, remember to check the following items so that the Synchronization Server is referencing the net-snmp installation:

- The shell PATH will reference the native implementation binaries. Adjust the PATH variable or invoke the net-snmp binaries explicitly.
- If the native daemon is not stopped, there will likely be port conflicts between the native daemon and the net-snmp daemon. Disable the native daemon or use distinct port numbers for each.

SNMP on AIX Security Considerations

On AgentX sub-agent-compliant systems, it is recommended to use `agentXSocket tcp:localhost:705` to configure the net-snmp master agent to allow connections only from sub-agents located on the same host. On AIX systems, it is possible to specify an external IP network interface (for example, `agentXSocket tcp:0.0.0.0:708` would listen on all external IP interfaces), which would allow the UnboundID® Synchronization Server to be located on a different host to the snmp master agent.

While it is possible to implement non-local sub-agents, administrators should understand the security risks that are involved with this configuration. Primarily, because there is no communication authentication or privacy between the UnboundID® Synchronization Server and the master agent. An eavesdropper might be able to listen in on the monitoring data sent by the UnboundID® Synchronization Server. Likewise, a rogue sub-agent might be able to connect to the master agent and provide false monitoring data or deny access to SNMP monitoring data.

In general, it is recommended that sub-agents be located on the same host as the master agent.

MIBS

The Synchronization Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the proxy server connects to, and statistics about the operations invoked by the proxy server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Synchronization Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.

- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the directory server and the directory proxy server, see the text files provided in the `resource/mib` directory below the server installation.

The Synchronization Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Chapter 10 Troubleshooting the Synchronization Server

The UnboundID Synchronization Server provides a highly-available background synchronization solution for all types of network configurations. However, problems can arise from issues in the Synchronization Server itself or from a supporting component, like the JVM, operating system, or hardware. The Synchronization Server provides tools to diagnose any problems quickly to determine the underlying cause and the best course of action to take towards a resolution.

This chapter provides information on how to perform this analysis to help ensure that the problem is resolved as quickly as possible. It targets cases in which the Synchronization Server is running on Solaris or Linux systems, but much of the information can be useful on other platforms.

This chapter presents the following information:

Topics:

- *About Synchronization Troubleshooting*
- *About the Troubleshooting Tools*
- *Troubleshooting Process Flow*
- *Using the Sync Log*
- *Troubeshooting Sync Failures*
- *Problems with the Metrics Engine Management Console: JVM Memory Issues*
- *Working with the Collect Support Data Tool*

About Synchronization Troubleshooting

The majority of synchronization problems involve issues around the connection state of the external servers and the synchronization of the data between the two endpoints. Administrators should check if the Synchronization Server properly failed over to another endpoint instance if the connection was down on the highest priority external server. Further, if the main Synchronization Server is down for any reason, administrators should check if the Synchronization Server properly failed over to another Synchronization Server instance.

When troubleshooting synchronization information, administrators must determine if the DN and attribute mappings were properly configured and if the information is properly being synchronized across the network. Typical scenarios involve checking for any entry sync failures and mapping issues.

About the Troubleshooting Tools

The Synchronization Server provides utilities to troubleshoot the synchronization state of your server and to locate the causes of any problems that have occurred. The following tools are available for diagnosing any problems and are located in the `<server-root>/bin` directory on UNIX or Linux systems, or the `<server-root>/bat` directory on Windows systems:

Table 23: Troubleshooting Tools

Tool	Description
status	The `status` tool provides a high-level view of the current operational state of the Synchronization Server and displays any recent alerts that have occurred in past 24 hours. You can specify the `--pipe-name` argument to restrict the output to a specific sync pipe.
ldap-diff	The `ldap-diff` tool can be used to compare one or more entries across two server end-points to determine any data sync issues.
ldapsearch	The `ldapsearch` tool is used to get the full entries from two different servers if you want to review the exact content of an entry from each server.
logs	The logs directory provides important logs that should be used to troubleshoot or monitor any issue with the Synchronization Server: ➢ **Sync log** provides information about the synchronization operations that occur within the server. Specifically, the Sync Log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization. The log also shows the entries that were involved in the synchronization process. ➢ **Sync Failed Operations Log** provides a list of synchronization operations that have failed for any reason. ➢ **Resync log** provides summaries or details of synchronized entries and any missing entries in the Sync Destination. ➢ **Error log** provides information about warnings, errors, or significant events that occur within the server. ➢ **Debug log** can provide detailed information, if enabled, about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database. ➢ **Access loggers** provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the UnboundID Sync Management console.

Tool	Description
	For more information, see Managing Logging and Alerts.
resync	The `resync` tool can be used to validate your sync classes and your data mappings from one endpoint to another (DN or attribute maps). The tool provides a dry-run mode that sees what could happen to data using an operation without actually affecting the data.
collect-support-data	The `collect-support-data` tool is used to aggregate the results of various support tools data for the UnboundID Support team to diagnose. For more information, see Working with the Collect Support Data Tool.

Troubleshooting Process Flow

The general troubleshooting flow involves checking the status of the Synchronization Server, and then looking at the log files for information. The general flow is as follows:

1. **Run Status**. Run the `status` command to get the synchronization state information for your synchronization network.
2. **Check the Sync Log**. Depending on the nature of the problem, check the sync log file to diagnose any potential problems.
3. **Check the Failed Operations Log**. If you believe that the issue is data synchronization-related, then check the `logs/sync-failed-ops.log` to look at the cause of an issue.
4. **Check Synchronization Server Error Logs**. If the issue is a connectivity problem related to the source or destination servers, check the Synchronization Server error logs and the external server error logs.
5. **Check Endpoint Server Logs**. Look at the access and error logs on the source and destination servers.
6. **Run Collect-Support-Data**. If the Synchronization Server is experiencing issues that require assistance from your authorized support provider, then run the **collect-support-data** tool right away while the server is up and running to gather as much information as possible.

Using the Sync Log

The Sync log, located in the logs directory (`<server-root>/logs/sync`), provides useful troubleshooting information on the type of operation that was processed or completed. Most log entries provide the following common elements in their messages:

Table 24: Sync Logs Elements

Sync Log Element	Description
category	Indicates the type of operation, which will always by SYNC.
severity	Indicates the severity type of the message: INFORMATION, MILD_WARNING, SEVERE_WARNING, MILD_ERROR, SEVERE_ERROR, FATAL_ERROR, DEBUG, or NOTICE.
msgID	Specifies the unique ID number assigned to the message.
op	Specifies the operation number specific to sync.
changeNumber	Specifies the change number from the source server assigned to the modification.
replicationCSN	Specifies the replication change sequence number from the source server.
replicaID	Specifies the replica ID from the source server if there are multiple backend databases.
pipe	Specifies the sync pipe that was used to sync this operation.

Sync Log Element	Description
msg	Displays the result of the sync operation.

Sync Log Example 1

The following example displays an informational message that a modification to an entry was detected on the source server.

```
$ tail -f logs/sync

[17/May/2010:15:46:19 -0500] category=SYNC severity=INFORMATION msgID=1893728293 op=14
changeNumber=15 replicationCSN=00000128A7E3C7D31E960000000F replicaID=7830 pipe="DS1 to
DS2" msg="Detected MODIFY of uid=user.993,ou=People,dc=example,dc=com at ldap://
server1.example.com:1389"
```

Sync Log Example 2

The next example shows a successful synchronization operation that resulted from a MODIFY operation on the source server and synchronized to the destination server.

```
[18/May/2010:13:54:04 -0500] category=SYNC severity=INFORMATION msgID=1893728306
op=701 changeNumber=514663 replicationCSN=00000128ACC249A31E960007DA67 replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Synchronized MODIFY of uid=user.698,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389 by modifying entry uid=user.698,
ou=People,dc=example,dc=com at ldap://server3.example.com:3389"
```

Sync Log Example 3

The next example shows a failed synchronization operation on a MODIFY operation from the source server that could not be synchronized on the destination server. The log displays the LDIF-formatted modification that failed, which came from a schema violation that resulted from an incorrect attribute mapping (telephoneNumber -> telephone) from the source to destination server.

```
[18/May/2010:11:29:49 -0500] category=SYNC severity=SEVERE_WARNING msgID=1893859389
op=71831 changeNumber=485590 replicationCSN=00000128AC3DE8D51E96000768D6
replicaID=7830 pipe="DS1 to DS2" class="DEFAULT" msg="Detected MODIFY of
uid=user.941,ou=People,dc=example,dc=com at ldap://server1.example.com:1389, but
failed to apply this change because: Failed to modify entry uid=user.941,
ou=People,dc=example,dc=com on destination 'server3.example.com:3389'.
Cause: LDAPException(resultCode=65(object class violation), errorMessage='
Entry uid=user.941,ou=People,dc=example,dc=com cannot be modified because the
resulting entry would have violated the server schema: Entry uid=user.941,ou=People,
dc=example,dc=com violates the Directory Server schema configuration because it
includes attribute telephone which is not allowed by any of theobjectclasses
defined in that entry') (id=1893859386 ResourceOperationFailedException.java:125
Build revision=6226). Details: Source change detail:

dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: 027167170433915
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: ds-update-time
ds-update-time:: AAABKKw96NU=
 Equivalent destination changes:
dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephone
```

```
telephone: 818002279103216
 Full source entry:
dn: uid=user.941,ou=People,dc=example,dc=com
objectClass: person
... (more output)
Mapped destination entry:
dn: uid=user.941,ou=People,dc=example,dc=com
telephone: 818002279103216
objectClass: person
objectClass: inetOrgPerson
... (more output) ...
```

Troubeshooting Sync Failures

While many Synchronization Server issues are deployment-related and are directly affected by the hardware, software, and network structure used in the synchronization topology, most sync failures usually fall into one of three categories:

- **Entry Already Exists**. Indicates that when an add operation was attempted on the destination server, an entry with the same DN already exists.
- **No Match Found**. Indicates that a match was not found at the destination based on the current sync classes and correlation rules (i.e., DN and attribute mapping). When this value has a high count, it is likely that there were correlation rule problems. For example, use bin/status and look for "No Match Found".
- **Failure at Resource**. Indicates that some other error happened during the sync process that does not fall into the above categories. Typically, these errors are communication problems with a source or destination server.

Statistics for these and numerous other types of errors are kept under the cn=monitor branch and can be viewed directly using the status command.

Troubleshooting "Entry Already Exists" Failures

The status utility provides a comprehensive view of your synchronization network and displays the operation statistics to diagnose any potential problems with the Synchronization Server or the external servers. If you see that there is a count for the Entry Already Exists statistic using the status tool, then verify the problem in the sync log. For example, the status tool displays the following information:

```
  --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Then verify the change by viewing the <server-root>/logs/sync file to see the specific operation, which could be due to someone manually adding the entry on the target server:

```
[18/May/2010:15:14:30 -0500] category=SYNC severity=SEVERE_WARNING msgID=1893859372
op=2 changeNumber=529277 replicationCSN=00000128AD0D9BA01E960008137D replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Detected ADD of uid=user.1001,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389, but cannot create this entry
at the destination because an equivalent entry already exists at ldap://server3.
example.com:3389. Details: Search using [search-criteria dn: uid=user.1001,ou=People,
dc=example,dc=com attrsToGet: [*, dn]] returned results; [uid=user.1001,ou=People,
dc=example,dc=com]. "
```

However, in the following example, a client attempted a MODIFY operation on an entry (uid=1234) on the source server, but the Synchronization Server could not find the entry on the destination server when it ran an initial search. The Synchronization Server then changed the MODIFY request to an ADD operation request to add the entry to the destination server. The ADD operation subsequently failed because an entry with the same DN already existed on the target server. In a case like this, the main problem could be due to an incorrectly-formed correlation rule (DN mapping) defined in the Sync Class used in the Sync Pipe.

```
[12/May/2010:00:00:53 -0500] category=SYNC severity=SEVERE_WARNING msgID=1893859389
op=2827888 changeNumber=5317162 replicationCSN=4bea4af3000b21140000 replic-
aID=8468,dc=example,dc=com pipe="DS1 to DS2" class=”FullSync" msg="Detected MODIFY of
uid=1234,ou=People,dc=example,dc=com at ldap://server1.example.com:389, but failed to
apply this change because: Failed to create entry uid=1234,ou=People,dc=example,
dc=com on destination ’server1:389'. Cause: LDAPException(resultCode=entry already
exists, errorMessage='The entry uid=1234,ou=People,dc=dest,dc=com cannot be added
because an entry with that name already exists') (id=1893859385)”
```

To Troubleshoot an "Entry Already Exists" Problem

1. Assuming that a possible DN mapping is ill-formed, you should first run the ldap-diff utility to compare the entries on the source and destination servers. Then look at the `ldap-diff` results with your mapping rules to see why the original search did not find a match.

   ```
   $ bin/ldap-diff \
   --outputLDIF config-difference.ldif \
   --baseDN "dc=example,dc=com" \
   --sourceHost server1.example.com \
   --targetHost server2.example.com \
   --sourcePort 1389 \
   --targetPort 3389 \
   --sourceBindDN "cn=Directory Manager" \
   --sourceBindPassword password \
   --searchFilter "(uid=1234)"
   ```

2. Next, look at the destination server access logs to verify the search and filters it used to find the entry. Typically, you will find that your key correlation attributes are out-of-sync, which is why the search failed.

3. If the mapping rule attributes are out-of-sync, then you need to determine why that happened. Review your sync classes and mapping rules, and use the information from the `ldap-diff` results to determine why a specific attribute may not be getting updated. Some questions to answer are as follows:

   - Do you have more than one sync class that the operation could be matched with?
   - If you use an "include-base-dn" or "include-filter" in your mapping rules, does this exclude this operation by mistake?
   - If you use an attribute map, are your mappings correct? Usually, the cause in these type of messages are the destination mapping attribute settings. For example, if you define a set of correlation attributes as follows: dn, mobile, accountNumber. And the accountNumber

changes for some reason, this will cause future operations on this entry to fail. To resolve this, you would either remove accountNumber from the rule, or add a second rule as follows: dn, mobile. The second rule will only be used if the search using the first set of attributes fails. In this case, the entry will be found and the accountNumber information will also be updated.

4. If you have deletes being synced, check to see if there was a previous delete of this entry that did not sync properly. In some cases, you will have to use simpler logic for deletes than other operations due to the available attributes in the change logs. This scenario could cause an entry to not be deleted for some reason, which would cause an issue when a new entry with the same DN is added later. You can then use this information with your mapping rules to see why the original search did not find a match.

5. Look at the destination directory server access logs to verify the search and filters it used to find the entry. Typically, you will find that your key attribute mappings are out-of-sync.

Troubleshooting "No Match Found" Failures

If you see that there is a count for the No Match Found statistic using the `status` tool, then verify the problem in the sync log. For example, the status tool displays the following information:

```
--- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Then verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation:

```
[12/May/2010:10:30:45 -0500] category=SYNC severity=MILD_WARNING msgID=1893793952
op=4159648 changeNumber=6648922 replicationCSN=4beadaf4002f21150000 replicaID=8469-
ou=test,dc=example,dc=com pipe="DS1 to DS2" class="Others" msg="Detected DELETE of
'uid=1234,ou=test,dc=example,dc=com' at ldap://server1.example.com:389,  but cannot
DELETE this entry at the destination because no matching entries were found at ldap://
server2.example.com:389. Details: Search using [search-criteria dn:
uid=1234,ou=test,dc=alu,dc=com filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-
717b93af) attrsToGet: [*, accountNumber, dn, entryuuid, mobile, nsUniqueId, object-
Class]] returned no results."
```

To Troubleshoot "No Match Found" Failures

1. First, test the search using the filter in the error message if displayed. For example, the sync log specifies `"filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-717b93af)"`. Use the `ldapsearch` tool to test the filter. Did this succeed? If yes, can you see anything in the your attribute mappings that would exclude this from working properly?

2. Next, test the search using the full DN as the base. For example, use ldapsearch with the full DN (`uid=1234,ou=People,dc=example,dc=com`). Did this succeed? If yes, then does the entry contain the attribute used in the mapping rule?

3. If the attribute is not in the entry, then determine if there is a reason why this attribute value was not synced in the first place. Look at the attribute mappings and the filters used in the sync classes.

Troubleshooting "Failed at Resource" Failures

If you see that there is a count for the "Failed at Resource" statistic using the `status` tool, then verify the problem in the sync log. For example, the `status` tool displays the following information:

```
--- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 0
No Match Found         : 0
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 1
Unexpected Exception   : 0
Total                  : 1
```

You will see this stat after a change has been detected at the source in any of the following cases:

- If the fetch of the full source entry fails. In this case, the entry exists but there is a connection failure, server down, timeout, etc.
- If the fetch of the destination entry fails or if the modification to the destination fails for an exceptional reason (but not for cases "Entry Already Exists," "Multiple Matches Found," "No Match Found").

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation. If you see any of the following resultCodes, then your server is seeing timeout errors:

- resultCode=timeout: errorMessage=A client-side timeout was encountered while waiting 60000ms for a search response from server server1.example.com:1389
- resultCode=timeout: errorMessage=An I/O error occurred while trying to read the response from the server
- resultCode=server down: errorMessage=An I/O error occurred while trying to read the response from the server
- resultCode=server down: errorMessage=The connection to server `server1.example.com:1389` was closed while waiting for a response to search request SearchRequest

- resultCode=object class violation: `errorMessage='Entry device=1234,dc=example,dc=com` violates the Directory Server schema configuration because it contains undefined object class

To Troubleshoot "Failed at Resource" Failures

With the Failure at Destination timeout errors, you can look at the following settings in the Synchronization Server to see of they need adjustments:

1. **For External Server Properties**. Check the `connect-timeout` property. This property specifies the maximum length of time to wait for a connection to be established before giving up and considering the server unavailable.

2. **For the Sync Destination/Sync Source Properties**. Check the `response-timeout` property. This property specifies the maximum length of time that an operation should be allowed to be blocked while waiting for a response from the server. A value of zero indicates that there should be no client-side timeout. In this case, the server's default will be used.

```
$ bin/dsconfig --no-prompt --port 389 --bindDN "cn=Directory Manager" \
--bindPassword password list-external-servers --property connect-timeout

External Server          : Type         : connect-timeout : response-timeout
-------------------------:--------------:-----------------:-----------------
server1.example.com:389  : sundsee-ds   : 10 s            : -
server2.example.com:389  : sundsee-ds   : 10 s            : -
server3.example.com:389  : unboundid-ds : 10 s            : -
server4.example.com:389  : unboundid-ds : 10 s            : -
```

3. For Sync Pipe Properties. Check the `max-operation-attempts`, `retry-backoff-initial-wait`, `retry-backoff-max-wait`, `retry-backoff-increase-by`, `retry-backoff-percentage-increase`. These Sync Pipe Properties provide tuning parameters that are used in conjunction with the timeout settings. When a sync pipe experiences an error, then it will use these settings to determine how often and quickly it will retry the operation.

```
$ bin/dsconfig --no-prompt list-sync-pipes \
--property max-operation-attempts --property retry-backoff-initial-wait \
--property retry-backoff-max-wait --property retry-backoff-increase-by \
--property retry-backoff-percentage-increase \
--port 389 --bindDN "cn=Directory Manager" --bindPassword password
```

Problems with the Metrics Engine Management Console: JVM Memory Issues

Console runs out of memory (PermGen). If you are running a Management Console for a UnboundID® Directory Server while also running a console for the UnboundID® Directory Proxy Server Management Console and an UnboundID® Synchronization Server Management Console, you may see a Java PermGen error as follows:

```
Exception in thread "http-bio-8080-exec-7" java.lang.OutOfMemoryError: PermGen Space
```

For a servlet container, such as Tomcat, you can specify additional arguments to pass to the JVM by creating a `bin/setenv.sh` file (or `setenv.bat` for Windows) that sets the

CATALINA_OPTS variable. The `startup.sh` script will automatically pick this up. For example:

```
#!/bin/bash
# The following may be modified to change JVM memory arguments.
MAX_HEAP_SIZE=512m
MIN_HEAP_SIZE=$MAX_HEAP_SIZE
MAX_PERM_SIZE=256m

CATALINA_OPTS="-Xmx${MAX_HEAP_SIZE} -Xms${MIN_HEAP_SIZE} -XX:MaxPermSize=
${MAX_PERM_SIZE}"
```

Working with the Collect Support Data Tool

The Synchronization Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for Solaris and Linux machines, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. For example, on Solaris Zone, configuration information is gathered using commands like `zonename` and `zoneadm`. However, the tool always tries to get the same information across all systems for the target Synchronization Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Server Commands Used in the Collect Support Data Tool

The following presents a summary of the data collectors that the `collect-support-data` tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

Table 25: Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
status	Runs `status -F` to show the full version information of the Synchronization Server (Unix, Windows).
server-state	Runs `server-state` to show the current state of the Synchronization Server process (Unix, Windows).
dsreplication status	Runs `dsreplication status` to show the status of the replicated topology (Unix, Windows). If the `--noReplicationStatus` option is used, the replication status information is not collected.

JDK Commands Used in the Collect-Support-Data Tool

Table 26: JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Solaris, Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Solaris, Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Solaris, Linux, Windows, Mac OS).
jinfo	Displays the Java configuration information for the Java process (Solaris, Linux, Windows, Mac OS).

Linux Commands Used in the collect-support-data Tool

Table 27: Linux Commands Used in the Collect-Support-Data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the `/var/logs/messages` directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: ➢ /proc/cpuinfo ➢ /proc/meminfo ➢ /etc/hosts ➢ /etc/nsswitch.conf ➢ /etc/resolv.conf
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

Solaris Commands Used in the collect-support-data Tool

Table 28: Solaris Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
zonename	Prints the name of the current zone.
zoneadm	Prints the name of the current configured in verbose mode.

Data Collector	Description
df	Prints the amount of available disk space for filesystems in 1024-byte units.
zfs	Prints basic ZFS information: dataset pool names, and their used, available, referenced, and mountpoint properties.
zpool	Print a zpool's status.
fmdump	Prints the log files managed by the Solaris Fault Manager.
prtconf	Prints the system configuration information.
iostat	Prints disk I/O and CPU utilization information.
prtdiag	Prints the system diagnostic information.
cat	Concatenates the following files and prints to standard output: ➢ /proc/cpuinfo ➢ /proc/meminfo ➢ /etc/hosts ➢ /etc/nsswitch.conf ➢ /etc/resolv.conf
tail	Displays the last few lines of a file. Tails the `/var/logs/messages` directory and the `/var/log/system.log` directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
patchadd	Prints the patches added to the system if any (Solaris, not OpenSolaris).
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
prstat	Prints resource usage.

AIX Commands Used in the collect-support-data Tool

Table 29: AIX Commands Used in the Collect-Support-Data Tool

Data Collector	Description
ulimit	Defines user and system resources.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
zonename	Prints the name of the current zone.
cat	Concatenates the following files and prints to standard output: ➢ /proc/cpuinfo ➢ /proc/meminfo ➢ /etc/hosts ➢ /etc/nsswitch.conf ➢ /etc/resolv.conf
vmstat	Prints information about virtual memory statistics.
alog	Prints the contents of the boot log file.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sar	Print the local activity of the server.
lparstat	Prints logical partition information and statistics.

Data Collector	Description
vmo	Prints the characteristics of one or more tunable parameters.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.

MacOS Commands Used in the Collect Support Data Tool

Table 30: MacOS Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the `/var/log/system.log` directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sample	Profiles a process during an interval.

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-Sync --pid 1234
   ```

3. Email the zip file to your Authorized Support Provider.