FORGEROCK®

Administration Guide
/ ForgeRock Directory Services 5

Latest update: 5.0.0

Mark Craig
Nemanja Lukić
Ludovic Poitou
Chris Ridd

Copyright © 2011-2017 ForgeRock AS.

Abstract

Guide to configuring and using ForgeRock® Directory Services features.

Table of Contents

Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

This guide shows you how to configure, maintain, and troubleshoot ForgeRock Directory Services software. ForgeRock Directory Services allow applications to access directory data:

- Over Lightweight Directory Access Protocol (LDAP)

- Using Directory Services Markup Language (DSML)

- Over Hypertext Transfer Protocol (HTTP) by using HTTP methods in the Representational State Transfer (REST) style

In reading and following the instructions in this guide, you will learn how to:

- Use ForgeRock Directory Services administration tools

- Manage ForgeRock Directory Services processes

- Import, export, backup, and restore directory data

- Configure ForgeRock Directory Services connection handlers for all supported protocols

- Configure administrative privileges and fine-grained access control

- Index directory data, manage schemas for directory data, and enforce uniqueness of directory data attribute values

- Configure data replication

- Implement password policies, pass-through authentication to another directory, password synchronization with Samba, account lockout, and account status notification

- Set resource limits to prevent unfair use of server resources

- Monitor servers through logs and alerts and over JMX

- Tune servers for best performance

- Secure server deployments

- Change server key pairs and public key certificates

- Move a server to a different system

- Troubleshoot server issues

1. Using This Guide

This guide is intended for system administrators who build, deploy, and maintain ForgeRock Directory Services for their organizations.

This guide starts with an introduction to directory services. The rest of this guide is written with the assumption that you have basic familiarity with the following topics:

- The client-server model of distributed computing

- Lightweight Directory Access Protocol (LDAP), including how clients and servers exchange messages

- Managing Java-based services on operating systems and application servers

- Using command-line tools and reading command-line examples written for UNIX/Linux systems

- Configuring network connections on operating systems

- Managing Public Key Infrastructure (PKI) used to establish secure connections

Depending on the features you use, you should also have basic familiarity with the following topics:

- Directory Services Markup Language (DSML), including how clients and servers exchange messages

- Hypertext Transfer Protocol (HTTP), including how clients and servers exchange messages

- Java Management Extensions (JMX) for monitoring services

- Simple Network Management Protocol (SNMP) for monitoring services

2. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

• ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

3. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

ix

Chapter 1
Understanding Directory Services

This chapter introduces directory concepts and server features. In this chapter you will learn:

- Why directory services exist and what they do well

- How data is arranged in directories that support Lightweight Directory Access Protocol (LDAP)

- How clients and servers communicate in LDAP

- What operations are standard according to LDAP and how standard extensions to the protocol work

- Why directory servers index directory data

- What LDAP schemas are for

- What LDAP directories provide to control access to directory data

- Why LDAP directory data is replicated and what replication does

- What Directory Services Markup Language (DSML) is for

- How HTTP applications can access directory data in the Representation State Transfer (REST) style

A directory resembles a dictionary or a phone book. If you know a word, you can look it up its entry in the dictionary to learn its definition or its pronunciation. If you know a name, you can look it up its entry in the phone book to find the telephone number and street address associated with the name. If you are bored, curious, or have lots of time, you can also read through the dictionary, phone book, or directory, entry after entry.

Where a directory differs from a paper dictionary or phone book is in how entries are indexed. Dictionaries typically have one index—words in alphabetical order. Phone books, too—names in alphabetical order. Directories' entries on the other hand are often indexed for multiple attributes, names, user identifiers, email addresses, and telephone numbers. This means you can look up a directory entry by the name of the user the entry belongs to, but also by their user identifier, their email address, or their telephone number, for example.

ForgeRock Directory Services are based on the Lightweight Directory Access Protocol (LDAP). Much of this chapter serves therefore as an introduction to LDAP. ForgeRock Directory Services also provide RESTful access to directory data, yet, as directory administrator, you will find it useful to

understand the underlying model even if most users are accessing the directory over HTTP rather than LDAP.

1.1. How Directories and LDAP Evolved

Phone companies have been managing directories for many decades. The Internet itself has relied on distributed directory services like DNS since the mid 1980s.

It was not until the late 1980s, however, that experts from what is now the International Telecommunications Union published the X.500 set of international standards, including Directory Access Protocol. The X.500 standards specify Open Systems Interconnect (OSI) protocols and data definitions for general purpose directory services. The X.500 standards were designed to meet the needs of systems built according to the X.400 standards, covering electronic mail services.

Lightweight Directory Access Protocol has been around since the early 1990s. LDAP was originally developed as an alternative protocol that would allow directory access over Internet protocols rather than OSI protocols, and be lightweight enough for desktop implementations. By the mid-1990s, LDAP directory servers became generally available and widely used.

Until the late 1990s, LDAP directory servers were designed primarily with quick lookups and high availability for lookups in mind. LDAP directory servers replicate data, so when an update is made, that update is applied to other peer directory servers. Thus, if one directory server goes down, lookups can continue on other servers. Furthermore, if a directory service needs to support more lookups, the administrator can simply add another directory server to replicate with its peers.

As organizations rolled out larger and larger directories serving more and more applications, they discovered that they needed high availability not only for lookups, but also for updates. Around the year 2000, directories began to support multi-master replication; that is, replication with multiple read-write servers. Soon thereafter, the organizations with the very largest directories started to need higher update performance as well as availability.

The OpenDJ code base began in the mid-2000s, when engineers solving the update performance issue decided the cost of adapting the existing C-based directory technology for high-performance updates would be higher than the cost of building a next generation, high performance directory using Java technology.

1.2. About Data In LDAP Directories

LDAP directory data is organized into entries, similar to the entries for words in the dictionary, or for subscriber names in the phone book. A sample entry follows:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
uid: bjensen
cn: Babs Jensen
cn: Barbara Jensen
facsimileTelephoneNumber: +1 408 555 1992
gidNumber: 1000
givenName: Barbara
homeDirectory: /home/bjensen
l: San Francisco
mail: bjensen@example.com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: posixAccount
objectClass: top
ou: People
ou: Product Development
roomNumber: 0209
sn: Jensen
telephoneNumber: +1 408 555 1862
uidNumber: 1076
```

Barbara Jensen's entry has a number of attributes, such as `uid: bjensen`, `telephoneNumber: +1 408 555 1862`, and `objectClass: posixAccount`. (The `objectClass` attribute type indicates which types of attributes are required and allowed for the entry. As the entries object classes can be updated online, and even the definitions of object classes and attributes are expressed as entries that can be updated online, directory data is extensible on the fly.) When you look up her entry in the directory, you specify one or more attributes and values to match. The directory server then returns entries with attribute values that match what you specified.

The attributes you search for are indexed in the directory, so the directory server can retrieve them more quickly. (Attribute values do not have to be strings. Some attribute values, like certificates and photos, are binary.)

The entry also has a unique identifier, shown at the top of the entry, `dn: uid=bjensen,ou=People,dc=example,dc=com`. DN is an acronym for distinguished name. No two entries in the directory have the same distinguished name. Yet, DNs are typically composed of case-insensitive attributes.

Sometimes distinguished names include characters that you must escape. The following example shows an entry that includes escaped characters in the DN:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=escape)"
dn: cn=DN Escape Characters \" # \+ \, \; \< = \> \\,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: DN Escape Characters
uid: escape
cn: DN Escape Characters " # + , ; < = > \
sn: " # + , ; < = > \
mail: escape@example.com
```

LDAP entries are arranged hierarchically in the directory. The hierarchical organization resembles a file system on a PC or a web server, often imagined as an upside-down tree structure, or a pyramid. The distinguished name consists of components separated by commas, `uid=bjensen,ou=People,dc=example,dc=com`. The names are little-endian. The components reflect the hierarchy of directory entries.

"Directory Data" shows the hierarchy as seen in the control panel.

Directory Data
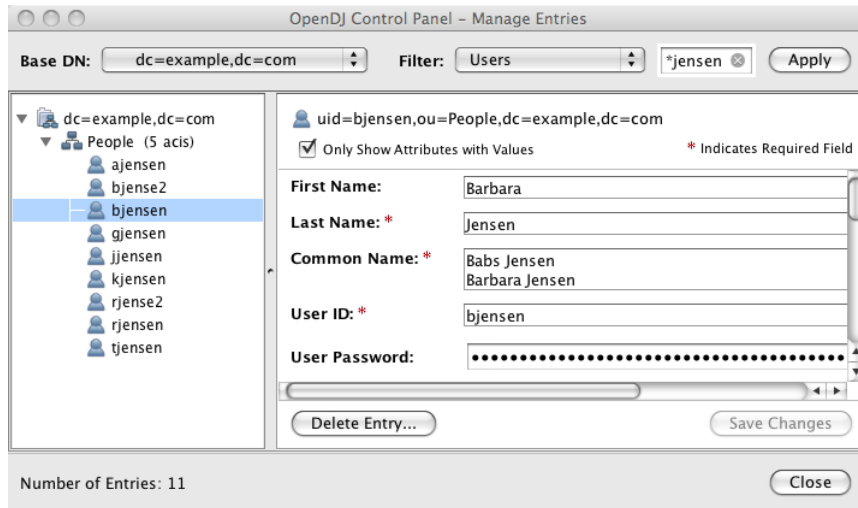
Barbara Jensen's entry is located under an entry with DN `ou=People,dc=example,dc=com`, an organization unit and parent entry for the people at Example.com. The `ou=People` entry is located under the entry with DN `dc=example,dc=com`, the base entry for Example.com. DC is an acronym for domain component. The directory has other base entries, such as `cn=config`, under which the configuration is accessible through LDAP. A directory can serve multiple organizations, too. You might find `dc=example,dc=com`, `dc=mycompany,dc=com`, and `o=myOrganization` in the same LDAP directory. Therefore, when you look up entries, you specify the base DN to look under in the same way you need to know whether to look

in the New York, Paris, or Tokyo phone book to find a telephone number. (The root entry for the directory, technically the entry with DN `""` (the empty string), is called the root DSE. It contains information about what the server supports, including the other base DNs it serves.

A directory server stores two kinds of attributes in a directory entry: *user attributes* and *operational attributes*. User attributes hold the information for users of the directory. All of the attributes shown in the entry at the outset of this section are user attributes. Operational attributes hold information used by the directory itself. Examples of operational attributes include `entryUUID`, `modifyTimestamp`, and `subschemaSubentry`. When an LDAP search operation finds an entry in the directory, the directory server returns all the visible user attributes unless the search request restricts the list of attributes by specifying those attributes explicitly. The directory server does not, however, return any operational attributes unless the search request specifically asks for them. Generally speaking, applications should change only user attributes, and leave updates of operational attributes to the server, relying on public directory server interfaces to change server behavior. An exception is access control instruction (`aci`) attributes, which are operational attributes used to control access to directory data.

1.3. About LDAP Client and Server Communication

In some client server communication, like web browsing, a connection is set up and then torn down for each client request to the server. LDAP has a different model. In LDAP the client application connects to the server and authenticates, then requests any number of operations, perhaps processing results in between requests, and finally disconnects when done.

The standard operations are as follows:

- Bind (authenticate). The first operation in an LDAP session usually involves the client binding to the LDAP server with the server authenticating the client. Authentication identifies the client's identity in LDAP terms, the identity which is later used by the server to authorize (or not) access to directory data that the client wants to lookup or change.

  If the client does not bind explicitly, the server treats the client as an anonymous client. An anonymous client is allowed to do anything that can be done anonymously. What can be done anonymously depends on access control and configuration settings. The client can also bind again on the same connection.

- Search (lookup). After binding, the client can request that the server return entries based on an LDAP filter, which is an expression that the server uses to find entries that match the request, and a base DN under which to search. For example, to look up all entries for people with the email address `bjensen@example.com` in data for Example.com, you would specify a base DN such as `ou=People,dc=example,dc=com` and the filter `(mail=bjensen@example.com)`.

- Compare. After binding, the client can request that the server compare an attribute value the client specifies with the value stored on an entry in the directory.

- Modify. After binding, the client can request that the server change one or more attribute values on an entry. Often administrators do not allow clients to change directory data, so allow appropriate access for client application if they have the right to update data.

- Add. After binding, the client can request to add one or more new LDAP entries to the server.

- Delete. After binding, the client can request that the server delete one or more entries. To delete an entry with other entries underneath, first delete the children, then the parent.

- Modify DN. After binding, the client can request that the server change the distinguished name of the entry. In other words, this renames the entry or moves it to another location. For example, if Barbara changes her unique identifier from `bjensen` to something else, her DN would have to change. For another example, if you decide to consolidate `ou=Customers` and `ou=Employees` under `ou=People` instead, all the entries underneath must change distinguished names.

  Renaming entire branches of entries can be a major operation for the directory, so avoid moving entire branches if you can.

- Unbind. When done making requests, the client can request an unbind operation to end the LDAP session.

- Abandon. When a request seems to be taking too long to complete, or when a search request returns many more matches than desired, the client can send an abandon request to the server to drop the operation in progress.

For practical examples showing how to perform the key operations using the command-line tools delivered with OpenDJ servers, read "*Performing LDAP Operations*" in the *Developer's Guide*.

1.4. About LDAP Controls and Extensions

LDAP has standardized two mechanisms for extending the operations directory servers can perform beyond the basic operations listed above. One mechanism involves using LDAP controls. The other mechanism involves using LDAP extended operations.

LDAP controls are information added to an LDAP message to further specify how an LDAP operation should be processed. For example, the Server-Side Sort request control modifies a search to request that the directory server return entries to the client in sorted order. The Subtree Delete request control modifies a delete to request that the server also remove child entries of the entry targeted for deletion.

One special search operation that OpenDJ servers support is Persistent Search. The client application sets up a Persistent Search to continue receiving new results whenever changes are made to data that is in the scope of the search, thus using the search as a form of change notification. Persistent Searches are intended to remain connected permanently, though they can be idle for long periods of time.

The directory server can also send response controls in some cases to indicate that the response contains special information. Examples include responses for entry change notification, password policy, and paged results.

For the list of supported LDAP controls, see "*LDAP Controls*" in the *Reference*.

LDAP extended operations are additional LDAP operations not included in the original standard list. For example, the Cancel Extended Operation works like an abandon operation, but finishes with a response from the server after the cancel is complete. The StartTLS Extended Operation allows a client to connect to a server on an unsecure port, but then starts Transport Layer Security negotiations to protect communications.

For the list of supported LDAP extended operations, see "*LDAP Extended Operations*" in the *Reference*.

1.5. About Indexes

As mentioned early in this chapter, directories have indexes for multiple attributes. In fact, by default OpenDJ does not let normal users perform searches that are not indexed, because such searches mean OpenDJ has to scan the entire directory looking for matches.

As directory administrator, part of your responsibility is making sure directory data is properly indexed. OpenDJ provides tools for building and rebuilding indexes, for verifying indexes, and also for evaluating how well they are working.

For help better understanding and managing indexes, read "*Indexing Attribute Values*".

1.6. About LDAP Schema

Some databases are designed to hold huge amounts of data for a particular application. Although such databases might support multiple applications, how their data is organized depends a lot on the particular applications served.

In contrast, directories are designed for shared, centralized services. Although the first guides to deploying directory services suggested taking inventory of all the applications that would access the directory, many current directory administrators do not even know how many applications use their services. The shared, centralized nature of directory services fosters interoperability in practice, and has helped directory services be successful in the long term.

Part of what makes this possible is the shared model of directory user information, and in particular the LDAP schema. LDAP schema defines what the directory can contain. This means that directory entries are not arbitrary data, but instead tightly codified objects whose attributes are completely predictable from publicly readable definitions. Many schema definitions are in fact standard. They are the same not just across a directory service but across different directory services.

At the same time, unlike some databases, LDAP schema and the data it defines can be extended on the fly while the service is running. LDAP schema is also accessible over LDAP. One attribute of every entry is its set of `objectClass` values. This gives you as administrator great flexibility in adapting your directory service to store new data without losing or changing the structure of existing data, and also without ever stopping your directory service.

For a closer look, see "*Managing Schema*".

1.7. About Access Control

In addition to directory schema, another feature of directory services that enables sharing is fine-grained access control.

As directory administrator, you can control who has access to what data when, how, where and under what conditions by using access control instructions (ACI). You can allow some directory operations and not others. You can scope access control from the whole directory service down to individual attributes on directory entries. You can specify when, from what host or IP address, and what strength of encryption is needed in order to perform a particular operation.

As ACIs are stored on entries in the directory, you can furthermore update access controls while the service is running, and even delegate that control to client applications. OpenDJ combines the strengths of ACIs with separate administrative privileges to help you secure access to directory data.

For more information, read "*Configuring Privileges and Access Control*".

1.8. About Replication

Replication in OpenDJ consists of copying each update to the directory service to multiple directory servers. This brings both redundancy, in the case of network partitions or of crashes, and also scalability for read operations. Most directory deployments involve multiple servers replicating together.

When you have replicated servers, all of which are writable, you can have replication conflicts. What if, for example, there is a network outage between two replicas, and meanwhile two different values are written to the same attribute on the same entry on the two replicas? In nearly all cases, OpenDJ replication can resolve these situations automatically without involving you, the directory administrator. This makes your directory service resilient and safe even in the unpredictable real world.

One perhaps counterintuitive aspect of replication is that although you do add directory *read* capacity by adding replicas to your deployment, you do not add directory *write* capacity by adding replicas. As each write operation must be replayed everywhere, the result is that if you have N servers, you have N write operations to replay.

Another aspect of replication to keep in mind is that it is "loosely consistent." Loosely consistent means that directory data will eventually converge to be the same everywhere, but it will not necessarily be the same everywhere right away. Client applications sometimes get this wrong when they write to a pool of load balanced directory servers, immediately read back what they wrote, and are surprised that it is not the same. If your users are complaining about this, either make sure their application always gets sent to the same server, or else ask that they adapt their application to work in a more realistic manner.

To get started with replication, see "*Managing Data Replication*".

1.9. About DSMLv2

Directory Services Markup Language (DSMLv2) v2.0 became a standard in 2001. DSMLv2 describes directory data and basic directory operations in XML format, so they can be carried in Simple Object Access Protocol (SOAP) messages. DSMLv2 further allows clients to batch multiple operations together in a single request, to be processed either in sequential order or in parallel.

OpenDJ provides support for DSMLv2 as a DSML gateway, which is a Servlet that connects to any standard LDAPv3 directory. DSMLv2 opens basic directory services to SOAP-based web services and service oriented architectures.

To set up DSMLv2 access, see "DSML Client Access".

1.10. About RESTful Access to Directory Services

OpenDJ can expose directory data as JSON resources over HTTP to REST clients, providing easy access to directory data for developers who are not familiar with LDAP. RESTful access depends on a configuration that describes how the JSON representation maps to LDAP entries.

Although client applications have no need to understand LDAP, OpenDJ's underlying implementation still uses the LDAP model for its operations. The mapping adds some overhead. Furthermore, depending on the configuration, individual JSON resources can require multiple LDAP operations. For example, an LDAP user entry represents `manager` as a DN (of the manager's entry). The same manager might be represented in JSON as an object holding the manager's user ID and full name, in which case OpenDJ must look up the manager's entry to resolve the mapping for the manager portion of the JSON resource, in addition to looking up the user's entry. As another example, suppose a large group is represented in LDAP as a set of 100,000 DNs. If the JSON resource is configured so that a member is represented by its name, then listing that resource would involve 100,000 LDAP searches to translate DNs to names.

A primary distinction between LDAP entries and JSON resources is that LDAP entries hold sets of attributes and their values, whereas JSON resources are documents containing arbitrarily nested objects. As LDAP data is governed by schema, almost no LDAP objects are arbitrary collections of data. (LDAP has the object class `extensibleObject`, but its use should be the exception rather than the rule.) Furthermore, JSON resources can hold arrays, ordered collections that can contain duplicates, whereas LDAP attributes are sets, unordered collections without duplicates. For most directory and identity data, these distinctions do not matter. You are likely to run into them, however, if you try to turn your directory into a document store for arbitrary JSON resources.

Despite some extra cost in terms of system resources, exposing directory data over HTTP can unlock directory services for a new generation of applications. The configuration provides flexible mapping, so that you can configure views that correspond to how client applications need to see directory data.

OpenDJ software also give you a deployment choice for HTTP access. You can deploy the REST to LDAP gateway, which is a Servlet that connects to any standard LDAPv3 directory, or you can activate the HTTP connection handler on an OpenDJ server to allow direct and more efficient HTTP and HTTPS access.

For examples showing how to use RESTful access, see "*Performing RESTful Operations*" in the *Developer's Guide*.

1.11. About Building Directory Services

This chapter is meant to serve as an introduction, and so does not even cover everything in this guide, let alone everything you might want to know about directory services.

When you have understood enough of the concepts to build the directory services that you want to deploy, you must still build a prototype and test it before you roll out shared, centralized services for your organization. Read "*Tuning Servers For Performance*" for a look at how to meet the service levels that directory clients expect.

Chapter 2
Administration Interfaces and Tools

This chapter covers OpenDJ administration tools. In this chapter you will learn to:

• Find and run OpenDJ control panel

• Find and run OpenDJ command-line tools

• Understand how command-line tools trust server certificates

• Use expressions in the server configuration file

OpenDJ server software installs with a cross-platform, Java Swing-based control panel for many day-to-day tasks. OpenDJ server software also installs command-line tools for configuration and management tasks.

This chapter is one of the few to include screen shots of the control panel. Most examples make use of the command-line tools. Once you understand the concepts and how to use the command-line tools, you only need to know where to start in the control panel to accomplish what you set out to do.

At a protocol level, administration tools and interfaces connect to servers through a different network port than that used to listen for traffic from other client applications.

This chapter takes a quick look at the tools for managing directory services.

2.1. Control Panel

OpenDJ control panel offers a GUI for managing both local and remote servers. You choose the server to manage when you start the control panel. The control panel connects to the administration server port, making a secure LDAPS connection.

The version of OpenDJ control panel must be the same as the target version of the OpenDJ server.

Start OpenDJ control panel by running the **control-panel** command, described in control-panel(1) in the *Reference*:

• (Linux, Solaris) Run **/path/to/opendj/bin/control-panel**.

• (Windows) Double-click `C:\path\to\opendj\bat\control-panel.bat`.

- (Mac OS X) Double-click `/path/to/opendj/bin/ControlPanel.app`.

When you log in to OpenDJ control panel, you authenticate over LDAP. This means that if users can run the control panel, they can use it to manage a running server. Yet, to start and stop the server process through OpenDJ control panel, you must start the control panel on the system where OpenDJ runs, as the user who owns the OpenDJ server files (such as the user who installed OpenDJ). In other words, the OpenDJ control panel does not do remote process management.

OpenDJ Control Panel

Down the left side of OpenDJ control panel shown in "OpenDJ Control Panel", notice what you can configure:

Directory Data

Directory data provisioning is typically not something you do by hand in most deployments. Usually entries are created, modified, and deleted through specific directory client applications. The Manage Entries window shown in "Manage Entries Window" can be useful in the lab as you design and test directory data, and if you modify individual ACIs or debug issues with particular entries.

Manage Entries Window

Additionally, the Directory Data list makes it easy to create a new base DN, and then import user data for the new base DN from LDAP Data Interchange Format (LDIF) files. You can also use the tools in the list to export user data to LDIF, and to backup and restore user data.

Schema

The Manage Schema window lets you browse and modify the rules that define how data is stored in the directory. You can add new schema definitions such as new attribute types and new object classes while the server is running, and the changes you make take effect immediately.

Indexes

The Manage Indexes window gives you a quick overview of all the indexes currently maintained for directory attributes. To protect your directory resources from being absorbed by costly searches on unindexed attributes, you may choose to keep the default behavior that prevents unindexed searches, and add only those indexes required by specific applications.

If the number of user data entries is smaller than the default resource limits, you can still perform what appear to be unindexed searches, meaning searches with filters for which no index appears to exist. That is because the `dn2id` index returns all user data entries without hitting a resource limit that would make the search unindexed.

OpenDJ control panel also allows you to verify and rebuild indexes, which you may have to do after an upgrade operation, or if you have reason to suspect index corruption.

Monitoring

The Monitoring list gives you windows to observe information about the system, the Java Virtual Machine (JVM) used, and indications about how the cache is used, whether the work queue has been filling up, as well as details about the database. You can also view the numbers and types of requests arriving over the connection handlers, and the current tasks in progress as well.

Runtime Options

This indicates where to change the runtime options for the server.

2.2. Server-Side Command-Line Tools

This section covers the tools installed with server software.

Before you try the examples in this guide, set your PATH to include the OpenDJ server tools. The location of the tools depends on the operating environment and on the packages used to install server software. "Paths To Administration Tools" indicates where to find the tools.

Paths To Administration Tools

OpenDJ running on...	OpenDJ installed from...	Default path to tools...
Apple Mac OS X, Linux distributions, Oracle Solaris	.zip	`/path/to/opendj/bin`
Linux distributions	.deb, .rpm	`/opt/opendj/bin`
Microsoft Windows	.zip	`C:\path\to\opendj\bat`
Oracle Solaris	SVR4	`/usr/opendj/bin`

You find the installation and upgrade tools, **setup**, and **upgrade**, in the parent directory of the other tools, as these tools are not used for everyday administration. For example, if the path to most tools is `/path/to/opendj/bin` you can find these tools in `/path/to/opendj`. For instructions on how to use the installation and upgrade tools, see the Installation Guide.

All OpenDJ command-line tools take the `--help` option.

All commands call Java programs and therefore involve starting a JVM.

"Tools and Server Constraints" indicates the constraints, if any, that apply when using a command-line tool with a server.

Tools and Server Constraints

Commands	Constraints
backendstat **create-rc-script**	These commands must be used with the local OpenDJ server in the same installation as the tools.

Commands	Constraints
encode-password **setup** **start-ds** **upgrade** **windows-service**	These commands are not useful with non-OpenDJ directory servers.
control-panel **dsconfig** **export-ldif** **import-ldif** **manage-account** **manage-tasks** **rebuild-index** **restore** **status** **stop-ds** **verify-index**	These commands must be used with an OpenDJ server having the same version as the command. These commands are not useful with non-OpenDJ directory servers.
dsreplication	With one exception, this command can be used with current and previous OpenDJ server versions. The one exception is the **dsreplication reset-change-number** subcommand, which requires OpenDJ server version 3.0.0 or later. This commands is not useful with other types of directory servers.
makeldif	This command depends on template files. The template files can make use of configuration files installed with an OpenDJ server under `config/MakeLDIF/`. The LDIF output can be used with any directory server.
base64 **ldapcompare** **ldapdelete** **ldapmodify** **ldappasswordmodify** **ldapsearch** **ldifdiff** **ldifmodify** **ldifsearch**	These commands can be used independently of an OpenDJ server, and so are not tied to a specific version.

The following list uses the UNIX names for the commands. On Windows all command-line tools have the extension .bat:

addrate

Measure add and delete throughput and response time.

For details, see addrate(1) in the *Reference*.

authrate

Measure bind throughput and response time.

For details, see authrate(1) in the *Reference*.

backendstat

Debug databases for pluggable backends.

For details, see backendstat(1) in the *Reference*.

backup

Back up or schedule backup of directory data.

For details, see backup(1) in the *Reference*.

base64

Encode and decode data in base64 format.

Base64-encoding represents binary data in ASCII, and can be used to encode character strings in LDIF, for example.

For details, see base64(1) in the *Reference*.

control-panel

Start OpenDJ Control Panel from the command-line.

create-rc-script (UNIX)

Generate a script you can use to start, stop, and restart the server either directly or at system boot and shutdown. Use **create-rc-script -f** *script-file*.

This allows you to register and manage an OpenDJ server as a service on UNIX and Linux systems.

For details, see create-rc-script(1) in the *Reference*.

dsconfig

The **dsconfig** command is the primary command-line tool for viewing and editing an OpenDJ configuration. When started without arguments, **dsconfig** prompts you for administration connection information. Once connected it presents you with a menu-driven interface to the server configuration.

Some advanced properties are not visible by default when you run the **dsconfig** command interactively. Use the `--advanced` option to access advanced properties.

When you pass connection information, subcommands, and additional options to **dsconfig**, the command runs in script mode and so is not interactive.

You can prepare **dsconfig** batch scripts by running the command with the `--commandFilePath` option in interactive mode, then reading from the batch file with the `--batchFilePath` option in script mode. Batch files can be useful when you have many **dsconfig** commands to run and want to avoid starting the JVM for each command.

Alternatively, you can read commands from standard input by using the `--batch` option.

For details, see dsconfig(1) in the *Reference*.

dsreplication

Configure data replication between directory servers to keep their contents in sync.

For details, see dsreplication(1) in the *Reference*.

encode-password

Encode a cleartext password according to one of the available storage schemes.

For details, see encode-password(1) in the *Reference*.

export-ldif

Export directory data to LDIF, the standard, portable, text-based representation of directory content.

For details, see export-ldif(1) in the *Reference*.

import-ldif

Load LDIF content into the directory, overwriting existing data. It cannot be used to append data to the backend database.

For details, see import-ldif(1) in the *Reference*.

ldapcompare

Compare the attribute values you specify with those stored on entries in the directory.

For details, see ldapcompare(1) in the *Reference*.

ldapdelete

Delete one entry or an entire branch of subordinate entries in the directory.

For details, see ldapdelete(1) in the *Reference*.

ldapmodify

Modify the specified attribute values for the specified entries.

For details, see ldapmodify(1) in the *Reference*.

ldappasswordmodify

Modify user passwords.

For details, see ldappasswordmodify(1) in the *Reference*.

ldapsearch

Search a branch of directory data for entries that match the LDAP filter you specify.

For details, see ldapsearch(1) in the *Reference*.

ldifdiff

Display differences between two LDIF files, with the resulting output having LDIF format.

For details, see ldifdiff(1) in the *Reference*.

ldifmodify

Similar to the **ldapmodify** command, modify specified attribute values for specified entries in an LDIF file.

For details, see ldifmodify(1) in the *Reference*.

ldifsearch

Similar to the **ldapsearch** command, search a branch of data in LDIF for entries matching the LDAP filter you specify.

For details, see ldifsearch(1) in the *Reference*.

makeldif

Generate directory data in LDIF based on templates that define how the data should appear.

The **makeldif** command is designed to help generate test data that mimics data expected in production, but without compromising real, potentially private information.

For details, see makeldif(1) in the *Reference*.

manage-account

Lock and unlock user accounts, and view and manipulate password policy state information.

For details, see manage-account(1) in the *Reference*.

manage-tasks

View information about tasks scheduled to run in the server, and cancel specified tasks.

For details, see manage-tasks(1) in the *Reference*.

modrate

Measure modification throughput and response time.

For details, see modrate(1) in the *Reference*.

rebuild-index

Rebuild an index stored in an indexed backend.

For details, see rebuild-index(1) in the *Reference*.

restore

Restore data from backup.

For details, see restore(1) in the *Reference*.

searchrate

Measure search throughput and response time.

For details, see searchrate(1) in the *Reference*.

start-ds

Start an OpenDJ server.

For details, see start-ds(1) in the *Reference*.

status

Display information about the server.

For details, see status(1) in the *Reference*.

stop-ds

Stop an OpenDJ server.

For details, see stop-ds(1) in the *Reference*.

verify-index

Verify that an index stored in an indexed backend is not corrupt.

For details, see verify-index(1) in the *Reference*.

windows-service (Windows)

Register and manage an OpenDJ server as a Windows Service.

For details, see windows-service(1) in the *Reference*.

2.3. How Command-Line Tools Trust Server Certificates

This section describes how OpenDJ command-line tools determine whether to trust server certificates.

When an OpenDJ command-line tool connects securely to a server, the server presents its digital certificate. The tool must then determine whether to trust the server certificate and continue negotiating the secure connection, or not to trust the server certificate and drop the connection.

An important part of trusting a server certificate is trusting the signing certificate of the party who signed the server certificate. The process is described in more detail in "About Certificates, Private Keys, and Secret Keys" in the *Security Guide*.

Put simply, a tool can automatically trust the server certificate if the tool's truststore contains the signing certificate. "Command-Line Tools and Truststores" indicates where to find the truststore. The signing certificate could be a CA certificate, or the server certificate itself if the certificate was self-signed.

When run in interactive mode, OpenDJ command-line tools can prompt you to decide whether to trust a server certificate not found in the truststore. When run in non-interactive mode, the tools must rely on the truststore.

Command-Line Tools and Truststores

Truststore Option	Truststore Used
None	The default truststore, `$HOME/.opendj/keystore`, where `$HOME` is the home directory of the user running the command-line tool. When you choose interactively to permanently trust a server certificate, the certificate is stored in this truststore.

`-P {trustStorePath}`
 	
 `--trustStorePath {trustStorePath}`
	Only the specified truststore is used.
	
	 In this case, the tool does not allow you to choose interactively to permanently trust an unrecognized server certificate.

2.4. Using Configuration Expressions

In the server configuration file, `config/config.ldif`, you can use Universal Expression Language (EL) expressions, to include variables values in configuration attributes.

Interface stability: Evolving

2.4.1. About Configuration Expressions

Configuration expressions introduce variables into the server configuration file, where you can use them in the values of configuration attributes. The server configuration framework evaluates expressions when it reads them.

To distinguish them from static values, expressions are preceded by a dollar sign and enclosed in braces, `${expression}`. An expression can be a value, a function call, or a complex expression with arithmetic, logical, relational, and conditional operators.

You can mix expressions and static values in configuration attributes. Inside the `${}` form, you can embed expressions. These capabilities are described below.

When evaluated, an expression must return the appropriate type for the configuration attribute, just like a static value. For example, `ds-cfg-listen-port` takes an integer, so if it is set using an expression, the attribute value resulting when the expression is evaluated must be an integer. If the type is wrong, the server fails to start due to a syntax error.

> **Important**
>
> The `config/config.ldif` file is not generally supported as a public interface to configure the server. The capability to use expressions in the configuration file is intended to make it easier to use the server in environments where the configuration is managed as a file, and some configuration attribute values must be determined dynamically.
>
> To use expressions, configure the server using the **setup** and **dsconfig** commands, then stop the server and edit the configuration file to replace attribute values with expressions.
>
> After that, if you use the **dsconfig** command to view or edit the server configuration, you do not see the expressions from the configuration file. Instead, when viewing the configuration you see the actual values. When editing the configuration, expressions are retained in the file unless you edit the attribute containing an expression. If you edit an attribute containing an expression, the your static value replaces the expression.

Configuration attributes that can be replaced by expressions have names that start with `ds-cfg-`. Some attributes in the configuration file are not strictly configuration attributes. For example, attributes like CN and `userPassword` must have concrete attribute values, rather than expressions.

2.4.2. Using Operators

EL supports the following operators, in order of precedence:

- Index property value: `[]`, `.`

- Change precedence of operation: `()`

- Unary negative: `-`

- Logical operations: `not`, `!`, `empty`

- Arithmetic operations: `*`, `/`, `div`, `%`, `mod`

- Binary arithmetic operations: `+`, `-`

- Relational operations: `<`, `>`, `<=`, `>=`, `lt`, `gt`, `le`, `ge`, `==`, `!=`, `eq`, `ne`

- Logical operations: `&&`, `and`, `||`, `or`

- Conditional operations: `?`, `:`

2.4.3. Reading System Properties and Environment Variables

Expressions can include Java system properties and environment variables.

For system properties, `${system['property']}` returns the value of *property*, or to an empty string if *property* is not set. For example, `${system['user.home']}` returns the home directory of the user running the server.

For environment variables, `${env['variable']}` returns the value of *variable*, or an empty string if *variable* is not set. For example, `${env['HOME']}` returns the home directory of the user running the server.

Values of system properties and environment variables are taken from the server process. In other words, values are set when you start the server process. To change the value, you must restart the server.

2.4.4. Using Built-In Functions

Expressions can include the following built-in functions:

`String decodeBase64(String value)`

Returns the base64-decoded string, or an empty string if the value is not valid Base64.

Example: `${decodeBase64('dGVzdA==')}` returns `test`.

`encodeBase64(String value)`

Returns the base64-encoded string, or an empty string if the input string is empty.

Example: `${encodeBase64('test')}` returns `dGVzdA==`.

`String read(String fileOrUrl)`

Returns the content of the specified local file or URL as a string, or an empty string if the file cannot be found or read.

Provide the parameter as an absolute file, a URL, or a path relative to the server base directory (`/path/to/opendj`).

Example: `${read('config/keystore.pin')}` returns the content of the `config/keystore.pin` file under the server base directory.

`Properties readProperties(String fileOrUrl)`

Returns the content of the specified local file or URL as a Properties object, or an empty string if the file cannot be found or read.

Provide the parameter as an absolute file, a URL, or a path relative to the server base directory (`/path/to/opendj`).

Example: `${readProperties('config/my.properties').port}` returns the value of the `port` property from the `config/my.properties` file under the server base directory.

`String trim(String string)`

Returns a copy of the string with leading and trailing whitespace removed.

Example: `${' Hello, world. '}` returns `Hello, world.`.

In addition to built-in functions, expressions can include static methods of the objects used in the expressions. For example, `${'my string'.toUpperCase()}` returns `MY STRING`.

2.4.5. Escaping, Embedding, and Concatenating Expressions

The backslash `\` character is the escape character. For example, to include the string `${true}` in an expression, write `\${true}`. To include a single backslash `\` character, write `${'\\'}`. To include a double backslash, write `${'\\\\'}`.

Expressions can be embedded, as in `${read(env['MY_FILE'])}`, which reads the content of the file specified by the environment variable, `MY_FILE`.

Expressions can be concatenated, as in `1${env['BASE_PORT']}`. If `BASE_PORT=389`, this expression returns `1389`.

2.4.6. Handling Sensitive Information

The actual values are visible when you read the configuration. As a result, if you use an expression to store sensitive information, the sensitive information is visible when reading the configuration.

For example, suppose the keystore PIN is stored in a file, and that this is stored in the configuration as follows:

```
ds-cfg-key-store-pin: ${read(config/keystore.pin)}
```

When the configuration is read, the server evaluates the expression and returns the actual value, in this case the PIN itself:

```
ds-cfg-key-store-pin: actual-keystore-PIN
```

For cases like this, take advantage of existing configuration attributes that prevent disclosure of sensitive information. For example, use the `ds-cfg-key-store-pin-file` attribute rather than the expression:

```
ds-cfg-key-store-pin-file: config/keystore.pin
```

Chapter 3
Managing Server Processes

This chapter covers starting and stopping OpenDJ servers. In this chapter you will learn to:

- Start, restart, and stop OpenDJ servers using OpenDJ command-line tools, OpenDJ control panel, or system service integration on Linux and Windows systems

- Understand how to recognize that an OpenDJ server is recovering from a crash or abrupt shutdown

- Configure whether an OpenDJ directory server loads data into cache at server startup before accepting client connections

3.1. Starting a Server

Use one of the following techniques:

- Use the **start-ds** command, described in start-ds(1) in the *Reference*:

```
$ start-ds
```

  Alternatively, you can specify the `--no-detach` option to start the server in the foreground.

- (Linux) If the OpenDJ server was installed from a .deb or .rpm package, then service management scripts were created at setup time.

  Use the **service opendj start** command:

```
centos# service opendj start
Starting opendj (via systemctl):                        [  OK  ]
```

```
ubuntu$ sudo service opendj start
$Starting opendj: > SUCCESS.
```

- (UNIX) Create an RC script by using the **create-rc-script** command, described in create-rc-script(1) in the *Reference*, and then use the script to start the server.

  Unless you run OpenDJ on Linux as root, use the `--userName` *userName* option to specify the user who installed OpenDJ:

```
$ sudo create-rc-script \
 --outputFile /etc/init.d/opendj \
 --userName opendj

$ sudo /etc/init.d/opendj start
```

For example, if you run OpenDJ on Linux as root, you can use the RC script to start the server at system boot, and stop the server at system shutdown:

```
$ sudo update-rc.d opendj defaults
update-rc.d: warning: /etc/init.d/opendj missing LSB information
update-rc.d: see <http://wiki.debian.org/LSBInitScripts>
 Adding system startup for /etc/init.d/opendj ...
   /etc/rc0.d/K20opendj -> ../init.d/opendj
   /etc/rc1.d/K20opendj -> ../init.d/opendj
   /etc/rc6.d/K20opendj -> ../init.d/opendj
   /etc/rc2.d/S20opendj -> ../init.d/opendj
   /etc/rc3.d/S20opendj -> ../init.d/opendj
   /etc/rc4.d/S20opendj -> ../init.d/opendj
   /etc/rc5.d/S20opendj -> ../init.d/opendj
```

- (Windows) Register OpenDJ as a Windows Service by using the **windows-service** command, described in windows-service(1) in the *Reference*, and then manage the service through Windows administration tools:

```
C:\path\to\opendj\bat> windows-service.bat --enableService
```

By default OpenDJ saves a compressed version of the server configuration used on successful startup. This ensures that the server provides a last known good configuration, which can be used as a reference or copied into the active configuration if the server fails to start with the current active configuration. It is possible, though not usually recommended, to turn this behavior off by changing the global server setting `save-config-on-successful-startup` to `false`.

3.2. Stopping a Server

Although OpenDJ servers are designed to recover from failure and disorderly shutdown, it is safer to shut the server down cleanly, because a clean shutdown reduces startup delays during which OpenDJ server attempts to recover database backend state, and prevents situations where OpenDJ server cannot recover automatically.

Follow these steps to shut down OpenDJ server cleanly:

To Stop a Server Cleanly

1. (Optional) If you are stopping a replicated server *permanently*, for example, before decommissioning the underlying system or virtual machine, first remove the server from the replication topology:

```
$ dsreplication \
 unconfigure \
 --unconfigureAll \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --adminPassword password \
 --trustAll \
 --no-prompt
```

    This step unregisters the server from the replication topology, effectively removing its replication configuration from other servers. This step must be performed before you decommission the system, because the server must connect to its peers in the replication topology.

2. Before shutting down the system where OpenDJ server is running, and before detaching any storage used for directory data, cleanly stop the server using one of the following techniques:

    *   Use the **stop-ds** command, described in stop-ds(1) in the *Reference*:

```
$ stop-ds
```

    *   (Linux) If an OpenDJ server was installed from a .deb or .rpm package, then service management scripts were created at setup time.

        Use the **service opendj stop** command:

```
centos# service opendj stop
Stopping opendj (via systemctl):                         [  OK  ]
```

```
ubuntu$ sudo service opendj stop
$Stopping opendj: ... > SUCCESS.
```

    *   (UNIX) Create an RC script, and then use the script to stop the server:

```
$ sudo create-rc-script \
 --outputFile /etc/init.d/opendj \
 --userName opendj

$ sudo /etc/init.d/opendj stop
```

- (Windows) Register OpenDJ as a Windows Service, and then manage the service through Windows administration tools:

```
C:\path\to\opendj\bat> windows-service.bat --enableService
```

Do not intentionally kill the OpenDJ server process unless the server is completely unresponsive.

When stopping cleanly, the server writes state information to database backends, and releases locks that it holds on database files.

3.3. Restarting a Server

Use one of the following techniques:

- Use the **stop-ds** command:

```
$ stop-ds --restart
```

- (Linux) If the OpenDJ server was installed from a .deb or .rpm package, then service management scripts were created at setup time.

   Use the **service opendj restart** command:

```
centos# service opendj restart
Restarting opendj (via systemctl):                        [  OK  ]
```

```
ubuntu$ sudo service opendj restart
$Stopping opendj: ... > SUCCESS.

$Starting opendj: > SUCCESS.
```

- (UNIX) Create an RC script, and then use the script to stop the server:

```
$ sudo create-rc-script \
 --outputFile /etc/init.d/opendj \
 --userName opendj

$ sudo /etc/init.d/opendj restart
```

- (Windows) Register OpenDJ as a Windows Service, and then manage the service through Windows administration tools:

```
C:\path\to\opendj\bat> windows-service.bat --enableService
```

3.4. Server Recovery

OpenDJ tends to show resilience when restarting after a crash or after the server process is killed abruptly. OpenDJ might have to replay the last few entries in a transaction log. Generally, OpenDJ returns to service quickly.

Database recovery messages are found in the database log file, such as `/path/to/opendj/db/userRoot/dj.log`.

The following example shows two example messages from the recovery log. The first message is written at the beginning of the recovery process. The second message is written at the end of the process:

```
[/path/to/opendj/db/userRoot]Recovery underway, found end of log
...
[/path/to/opendj/db/userRoot]Recovery finished: Recovery Info ...
```

Directory data cached in memory is lost during a crash. Loading directory data into memory when the server starts can take time. OpenDJ directory servers start accepting client requests before this process is complete.

Chapter 4
Managing Directory Data

This chapter covers management of LDAP Data Interchange Format (LDIF) data. In this chapter you will learn to:

- Generate test LDIF data

- Import and export LDIF data

- Perform searches and modifications on LDIF files with command-line tools

- Create and manage database backends to house directory data imported from LDIF

- Delete database backends

LDIF provides a mechanism for representing directory data in text format. LDIF data is typically used to initialize directory databases, but also may be used to move data between different directories that cannot replicate directly, or even as an alternative backup format.

4.1. Generating Test Data

When you install OpenDJ, you have the option of importing sample data that is generated during the installation. This procedure demonstrates how to generate LDIF by using the **makeldif** command, described in makeldif(1) in the *Reference*.

To Generate Test LDIF Data

The **makeldif** command uses templates to provide sample data. Default templates are located in the `/path/to/opendj/config/MakeLDIF/` directory. The `example.template` file can be used to create a suffix with entries of the type `inetOrgPerson`. You can do the equivalent in OpenDJ control panel (Directory Data > New Base DN... > Import Automatically Generated Example Data).

1. Write a file to act as the template for your generated LDIF.

   The resulting test data template depends on what data you expect to encounter in production. Base your work on your knowledge of the production data, and on the sample template, `/path/to/opendj/config/MakeLDIF/example.template`, and associated data.

   See makeldif.template(5) in the *Reference* for reference information about template files.

2. Create additional data files for the content in your template to be selected randomly from a file, rather than generated by an expression.

   Additional data files are located in the same directory as your template file.

3. Decide whether you want to generate the same test data each time you run the **makeldif** command with your template.

   If so, provide the same `randomSeed` integer each time you run the command.

4. Before generating a very large LDIF file, make sure you have enough space on disk.

5. Run the **makeldif** command to generate your LDIF file.

   The following command demonstrates use of the example MakeLDIF template:

```
$ makeldif \
 --outputLDIF generated.ldif \
 --randomSeed 42 \
 /path/to/opendj/config/MakeLDIF/example.template
...LDIF processing complete....
```

4.2. Importing and Exporting Data

You can use OpenDJ control panel to import data (Directory Data > Import LDIF) and to export data (Directory Data > Export LDIF). The following procedures demonstrate how to use the **import-ldif** and **export-ldif** commands, described in import-ldif(1) in the *Reference* and export-ldif(1) in the *Reference*.

To Import LDIF Data

The most efficient method of importing LDIF data is to take the OpenDJ server offline. Alternatively, you can schedule a task to import the data while the server is online.

> **Note**
>
> Importing from LDIF overwrites all data in the target backend with entries from the LDIF data.

1. (Optional) If you do not want to use the default `userRoot` backend, create a new backend for your data.

   See "Creating a New Database Backend" for details.

2. The following example imports `dc=example,dc=org` data into the `userRoot` backend, overwriting existing data:

- If you want to speed up the process—for example because you have millions of directory entries to import—first shut down the server, and then run the **import-ldif** command:

```
$ stop-ds --quiet
$ import-ldif \
 --offline \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --ldifFile generated.ldif
```

- If not, schedule a task to import the data while online:

```
$ start-ds --quiet
$ import-ldif \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --ldifFile generated.ldif \
 --trustAll
```

The import task is scheduled through a secure connection to the administration port, by default `4444`. You can schedule the import task to start at a particular time using the `--start` option.

The `--trustAll` option trusts all SSL certificates, such as a self-signed OpenDJ server certificate.

3. If the server is replicated with other servers, initialize replication again after the successful import.

For details see "Initializing Replicas".

Initializing replication overwrites data in the remote servers in the same way that import overwrites existing data with LDIF data.

To Export LDIF Data

The following examples export `dc=example,dc=org` data from the `userRoot` backend:

1. To expedite export, shut down the server and then use the **export-ldif** command:

```
$ stop-ds --quiet
$ export-ldif \
 --offline \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif
```

2. To export the data while online, leave the server running and schedule a task:

```
$ export-ldif \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif \
 --start 0 \
 --trustAll
```

The `--start 0` option tells OpenDJ directory server to start the export task immediately.

You can specify a time for the task to start using the format *yyyymmddHHMMSS*. For example, `20250101043000` specifies a start time of 4:30 AM on January 1, 2025.

If the server is not running at the time specified, it attempts to perform the task after it restarts.

4.3. Other Tools For Working With LDIF Data

This section demonstrates the command-line tools for working with LDIF:

- ldifdiff, demonstrated in "Comparing LDIF Files".

- ldifmodify, demonstrated in "Updating LDIF".

- ldifsearch, demonstrated in "Searching LDIF".

4.3.1. Searching LDIF

The **ldifsearch** command is to LDIF files what the **ldapsearch** command is to directory servers:

```
$ ldifsearch \
 --baseDN dc=example,dc=com \
 generated.ldif \
 "(sn=Grenier)" \
 uid
dn: uid=user.4630,ou=People,dc=example,dc=com
uid: user.4630
```

The *ldif-file* replaces the `--hostname` and `--port` options used to connect to an LDAP directory. Otherwise, the command syntax and LDIF output is familiar to **ldapsearch** users.

4.3.2. Updating LDIF

The **ldifmodify** command lets you apply changes to LDIF files, generating a new, changed version of the original file:

```
$ cat changes.ldif
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
.
-
replace: initials
initials: ZZZ


$ ldifmodify \
 --outputLDIF new.ldif \
 generated.ldif \
 changes.ldif
```

The resulting target LDIF file is approximately the same size as the source LDIF file, but the order of entries in the file is not guaranteed to be identical.

4.3.3. Comparing LDIF Files

The **ldifdiff** command reports differences between two LDIF files in LDIF format:

```
$ ldifdiff generated.ldif new.ldif
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
delete: description
description: This is the description for Aaccf Amar
.
-
add: description
description: New description
.
-
delete: initials
initials: AAA
-
add: initials
initials: ZZZ
-
```

The **ldifdiff** command reads both files into memory, and constructs tree maps to perform the comparison. The command is designed to work with small files and fragments, and can quickly run out of memory when calculating differences between large files.

4.3.4. Using Standard Input With the LDIF Tools

For each LDIF tool, a double dash, `--`, signifies the end of command options, after which only trailing arguments are allowed. To indicate standard input as a trailing argument, use a bare dash, `-`, after the double dash.

How bare dashes can be used after a double dash depends on the tool:

ldifdiff

The bare dash can replace either the source LDIF file, or the target LDIF file argument.

To take source LDIF from standard input, use the following construction:

```
ldifdiff [options] -- - target.ldif
```

To take target LDIF from standard input, use the following construction:

```
ldifdiff [options] -- source.ldif -
```

ldifmodify

The bare dash can replace either the source LDIF file, or the changes LDIF file arguments.

To take source LDIF from standard input, use the following construction:

```
ldifmodify [options] -- - changes.ldif [changes.ldif ...]
```

To take changes LDIF from standard input, use the following construction:

```
ldifmodify [options] -- source.ldif -
```

ldifsearch

The bare dash can be used to take the source LDIF from standard input by using the following construction:

```
ldifsearch [options] -- - filter [attributes ...]
```

4.4. About Database Backends

OpenDJ directory server stores data in a *backend*. A backend is a private server repository that can be implemented in memory, as a file, or as an embedded database.

Database backends are designed to hold large amounts of user data. OpenDJ servers have tools for backing up and restoring database backends, as described in "*Backing Up and Restoring Data*". By default, an OpenDJ directory server stores user data in a database backend named `userRoot`.

When installing the server and importing user data, and when creating a database backend, you choose the backend type. OpenDJ directory servers use JE backends for local data.

The JE backend type is implemented using B-tree data structures. It stores data as key-value pairs, which is different from the relational model exposed to clients of relational databases.

A JE backend stores data on disk using append-only log files with names like `number.jdb`. The JE backend writes updates to the highest-numbered log file. The log files grow until they reach a specified size (default: 100 MB). When the current log file reaches the specified size, the JE backend creates a new log file.

To avoid an endless increase in database size on disk, JE backends clean their log files in the background. A cleaner thread copies active records to new log files. Log files that no longer contain active records are deleted.

By default, JE backends let the operating system potentially cache data for a period of time before flushing the data to disk. This setting trades full durability with higher disk I/O for good performance with lower disk I/O. With this setting, it is possible to lose the most recent updates that were not yet written to disk in the event of an underlying OS or hardware failure. You can modify this behavior by changing the advanced configuration settings for the JE backend.

When a JE backend is opened, it recovers by recreating its B-tree structure from its log files. This is a normal process, one that allows the backend to recover after an orderly shutdown or after a crash.

Due to the cleanup processes, JE backends can be actively writing to disk even when there are no pending client or replication operations. To back up a server using a file system snapshot, you must *stop the server before taking the snapshot*.

4.5. Creating a New Database Backend

You can create new backends using the **dsconfig create-backend** command, described in the *Server Configuration Reference*. When you create a backend, choose the type of backend that fits your purpose.

The following example creates a database backend named `exampleOrgBackend`. The backend is of type `je`, which relies on a JE database for data storage and indexing:

```
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name exampleOrgBackend \
 --type je \
 --set enabled:true \
 --set base-dn:dc=example,dc=org \
 --set db-cache-percent:25 \
 --trustAll \
 --no-prompt
```

Notice the setting `db-cache-percent:25`. This says to allocate 25% of memory available to the JVM to the new backend's database cache. The default setting for `db-cache-percent` allocates 50%. When creating a new database backend, take care to keep the total memory allocated to all database caches lower than the total memory available to the JVM. As an alternative to `db-cache-percent`, you can use `db-cache-size`. The `db-cache-size` value is a specific amount of memory, such as `2 GB`.

After creating the backend, you can view the settings as in the following example:

```
$ dsconfig \
 get-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name exampleOrgBackend \
 --trustAll \
 --no-prompt
Property                 : Value(s)
-------------------------:---------------------
backend-id               : exampleOrgBackend
base-dn                  : "dc=example,dc=org"
cipher-key-length        : 128
cipher-transformation    : AES/CBC/PKCS5Padding
compact-encoding         : true
confidentiality-enabled  : false
db-cache-percent         : 25
db-cache-size            : 0 b
db-directory             : db
enabled                  : true
writability-mode         : enabled
```

Alternatively, you can create a new backend in OpenDJ control panel (Directory Data > New Base DN > Backend > New Backend: *backend-name*).

When you create a new backend using the **dsconfig** command, OpenDJ directory servers create the following indexes automatically:

`aci` presence
`ds-sync-conflict` equality
`ds-sync-hist` ordering
`entryUUID` equality
`objectClass` equality

You can create additional indexes as described in "Configuring and Rebuilding Indexes".

4.6. Splitting Data Across Multiple Backends

In some cases, such as directory services with subtree replication, you might choose to split directory data across multiple backends.

In the example in this section, the `userRoot` backend holds all data under `dc=example,dc=com` except `ou=People,dc=example,dc=com`. A separate `peopleRoot` backend holds data under `ou=People,dc=example,dc=com`. Replication for these backends is configured separately as described in "Subtree Replication".

The following example assumes you perform the steps when initially setting up the directory data. It uses the sample data from :

```
# Create a backend for the data not in the sub-suffix:
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --type je \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --set db-cache-percent:25 \
 --trustAll \
 --no-prompt

# Create a backend for the data in the sub-suffix:
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name peopleRoot \
 --type je \
 --set enabled:true \
 --set base-dn:ou=People,dc=example,dc=com \
```

```
  --set db-cache-percent:25 \
  --trustAll \
  --no-prompt

# Import data not in the sub-suffix:
$ import-ldif \
  --hostname opendj.example.com \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --backendID userRoot \
  --excludeBranch ou=People,dc=example,dc=com \
  --ldifFile Example.ldif \
  --trustAll

# Import data in the sub-suffix:
$ import-ldif \
  --hostname opendj.example.com \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --backendID peopleRoot \
  --includeBranch ou=People,dc=example,dc=com \
  --ldifFile Example.ldif \
  --trustAll
```

If, unlike the example, you must split an existing backend, follow these steps:

1. Create the new backend.

2. Export the data for the backend.

3. Import the data for the sub-suffix into the new backend using the `--includeBranch` option.

4. Delete all data under the sub-suffix from the old backend.

   For an example, see "Delete: Removing a Subtree" in the *Developer's Guide*.

When you split an existing backend, you must handle the service interruption that results when you move the sub-suffix data out of the original backend. If there is not a maintenance window where you can bring the service down in order to move the data, consider alternatives for replaying changes applied between the time that you exported the data and the time that you retired the old sub-suffix.

4.7. Encrypting Directory Data

An OpenDJ directory server can encrypt directory data before storing it in a database backend on disk, keeping the data confidential until it is accessed by a directory client.

Data encryption is useful for at least the following cases:

Ensuring Confidentiality and Integrity

Encrypted directory data is confidential, remaining private until decrypted with a proper key.

Encryption ensures data integrity at the moment it is accessed. An OpenDJ directory cannot decrypt corrupted data.

Protection on a Shared Infrastructure

When you deploy directory services on a shared infrastructure you relinquish full and sole control of directory data.

For example, if an OpenDJ directory server runs in the cloud, or in a data center with shared disks, the file system and disk management are not under your control.

Data confidentiality and encryption come with the following trade-offs:

Equality Indexes Limited to Equality Matching

When an equality index is configured without confidentiality, the values can be maintained in sorted order. A non-confidential, cleartext equality index can therefore be used for searches that require ordering and searches that match an initial substring.

An example of a search that requires ordering is a search with a filter `"(cn<=App)"`. The filter matches entries with `commonName` up to those starting with `App` (case-insensitive) in alphabetical order.

An example of a search that matches an initial substring is a search with a filter `"(cn=A*)"`. The filter matches entries having a `commonName` that starts with `a` (case-insensitive).

In an equality index with confidentiality enabled, the OpenDJ directory server no longer sorts cleartext values. As a result, you must accept that ordering and initial substring searches are unindexed.

Performance Impact

Encryption and decryption requires more processing than handling cleartext values.

Encrypted values also take up more space than cleartext values.

Replication Configuration Before Encryption

A directory server provides data confidentiality without requiring you to supply a key for encryption and decryption. It encrypts the data using a symmetric key stored under `cn=admin data` in the admin-backend. The symmetric key is encrypted in turn with the server's public key also stored there. When multiple servers are configured to replicate data as described in "*Managing Data Replication*", the servers replicate the keys as well, allowing any server replica to decrypt any other replica's encrypted data.

The directory server generates a secret key the first time it must encrypt data. That key is then shared across the replication topology as described above, or until it is marked as compromised. (For details regarding compromised keys, see "Handling Compromised Keys".)

When you configure replication, the source server overwrites `cn=admin data` in the destination server. This data includes any secret keys stored there by the destination server.

Therefore, if you configure data confidentiality before replication, the destination server's keys disappear when you configure replication. The destination server can no longer decrypt any of its data.

To prevent this problem, always configure replication before configuring data confidentiality.

As explained in "Protect OpenDJ Server Files", an OpenDJ directory server does not encrypt directory data by default. This means that any user with system access to read directory files can potentially access directory data in cleartext.

You can verify what a system user could see with read access to backend database files by using the **backendstat dump-raw-db** command. The **backendstat** subcommands **list-raw-dbs** and **dump-raw-db** help you list and view the low-level databases within a backend. Unlike the output of other subcommands, the output of the **dump-raw-db** subcommand is neither decrypted nor formatted for readability. Instead, you can see values as they are stored in the backend file.

In a backend database, the `id2entry` index holds LDIF representations of directory entries. For a database that is not encrypted, the corresponding low-level database shows the cleartext strings, as is evident in the following example:

```
$ stop-ds --quiet
$ backendstat list-raw-dbs --backendId userRoot
...
/dc=com,dc=example/id2entry...
...
$ backendstat \
 dump-raw-db \
 --backendId userRoot \
 --dbName /dc=com,dc=example/id2entry
...
Key (len 8):
 00 00 00 00 00 00 00 1E                          ........
 Value (len 437):
 02 00 81 B1 03 01 06 27 75 69 64 3D 62 6A 65 6E  .......'uid=bjen
 73 65 6E 2C 6F 75 3D 50 65 6F 70 6C 65 2C 64 63  sen,ou=People,dc
 3D 65 78 61 6D 70 6C 65 2C 64 63 3D 63 6F 6D 01  =example,dc=com.
 06 11 01 08 01 13 62 6A 65 6E 73 65 6E 40 65 78  ......bjensen@ex
 61 6D 70 6C 65 2E 63 6F 6D 01 09 01 04 30 32 30  ample.com....020
 39 01 16 01 0C 65 6E 2C 20 6B 6F 3B 71 3D 30 2E  9....en, ko;q=0.
 38 01 10 01 29 75 69 64 3D 74 72 69 67 64 65 6E  8...)uid=trigden
 2C 20 6F 75 3D 50 65 6F 70 6C 65 2C 20 64 63 3D  , ou=People, dc=
 65 78 61 6D 70 6C 65 2C 64 63 3D 63 6F 6D 01 04  example,dc=com..
 02 13 50 72 6F 64 75 63 74 20 44 65 76 65 6C 6F  ..Product Develo
 70 6D 65 6E 74 06 50 65 6F 70 6C 65 01 0B 01 07  pment.People....
 42 61 72 62 61 72 61 01 0C 01 0F 2B 31 20 34 30  Barbara....+1 40
 38 20 35 35 35 20 31 38 36 32 01 0D 01 06 4A 65  8 555 1862....Je
 6E 73 65 6E 01 07 02 0E 42 61 72 62 61 72 61 20  nsen....Barbara
 4A 65 6E 73 65 6E 0B 42 61 62 73 20 4A 65 6E 73  Jensen.Babs Jens
 65 6E 01 0E 01 0D 2F 68 6F 6D 65 2F 62 6A 65 6E  en..../home/bjen
 73 65 6E 01 0F 01 0F 2B 31 20 34 30 38 20 35 35  sen....+1 408 55
 35 20 31 39 39 32 01 11 01 04 31 30 30 30 01 12  5 1992....1000..
 01 2E 7B 53 53 48 41 7D 33 45 66 54 62 33 70 37  ..{SSHA}3EfTb3p7
 71 75 6F 75 73 4B 35 34 2B 41 4F 34 71 44 57 6C  quousK54+AO4qDWl
 56 33 4F 39 54 58 48 57 49 4A 49 32 4E 41 3D 3D  V3O9TXHWIJI2NA==
 01 13 01 04 31 30 37 36 01 05 01 14 4F 72 69 67  ....1076....Orig
```

```
 69 6E 61 6C 20 64 65 73 63 72 69 70 74 69 6F 6E    inal description
 01 14 01 07 62 6A 65 6E 73 65 6E 01 15 01 0D 53    ....bjensen....S
 61 6E 20 46 72 61 6E 63 69 73 63 6F 01 01 02 01    an Francisco....
 24 38 38 37 37 33 32 65 38 2D 33 64 62 32 2D 33    $887732e8-3db2-3
 31 62 62 2D 62 33 32 39 2D 32 30 63 64 36 66 63    1bb-b329-20cd6fc
 65 63 63 30 35                                     ecc05
...
```

To maintain data confidentiality on disk, you must configure it explicitly. In addition to preventing read access by other users as described in "Setting Up a System Account for an OpenDJ Server", you can configure confidentiality for database backends. When confidentiality is enabled for a backend, an OpenDJ directory server encrypts entries before storing them in the backend.

> **Important**
>
> Encrypting stored directory data does not prevent it from being sent over the network in the clear.
>
> Apply the suggestions in "*Securing Network Connections*" in the *Security Guide* to protect data sent over the network.

Enable backend confidentiality with the default encryption settings as shown in the following example that applies to the `userRoot` backend:

```
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set confidentiality-enabled:true \
 --no-prompt \
 --trustAll
```

After confidentiality is enabled, entries are encrypted when next written. That is, an OpenDJ directory server does not automatically rewrite all entries in encrypted form. Instead, it encrypts each entry on update, for example, when a user updates their entry or when you import data.

If you import the data again after enabling confidentiality, you can see with the **backendstat dump-raw-db** command that the low-level database for `id2index` no longer contains cleartext:

```
$ stop-ds --quiet
$ import-ldif \
 --offline \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --ldifFile Example.ldif
$ backendstat \
 dump-raw-db \
 --backendId userRoot \
 --dbName /dc=com,dc=example/id2entry
```

```
...
Key (len 8):
  00 00 00 00 00 00 00 C2                          ........
Value (len 437):
  02 02 81 82 01 C4 95 87 5B A5 2E 47 97 80 23 F4  ........[..G..#.
  CE 5D 93 25 97 D4 13 F9 0A A3 A8 31 9A D9 7A 70  .].%.......1..zp
  FE 3E AC 9D 64 41 EB 7B D5 7F 7E B8 B7 74 52 B8  .>..dA.{.^?~..tR.
  C7 7F C8 79 19 46 7D C5 5D 5B 83 9C 5B 9F 85 28  .^?.y.F}.][..[..(
  83 A2 5F A0 C1 B1 09 FC 2F E3 D8 82 4A AA 8B D9  .._...../...J...
  78 43 34 50 AE A1 52 88 5B 70 97 D2 E1 EA 87 CA  xC4P..R.[p.....
  3B 4D 07 DC F9 F8 30 BB D2 76 51 C8 75 FF FA 80  ;M....0..vQ.u...
  77 E1 6A 8B 5B 8F DA A4 F4 0B B5 20 56 B3 19 19  w.j.[...... V...
  22 D8 9D 38 04 E3 4D 94 A7 99 4B 81 16 AD 88 46  "..8..M...K....F
  FC 3F 7E 78 66 B8 D1 E9 86 A0 F3 AC B6 68 0D A9  .?~xf........h..
  9A A7 3C 30 40 37 97 4E 90 DD 63 16 8E 11 0F 5E  ..<0@7.N..c....^
  9D 5B 86 90 AF 4E E2 1F 9E 70 73 14 0A 11 5C DB  .[...N...ps...\.
  B7 BC B8 A9 31 3F 74 8D 0A 9F F4 6C E1 B0 36 78  ....1?t....l..6x
  F0 5A 5E CD 7C B3 A2 36 66 8E 88 86 A0 8B 9A 77  .Z^.|..6f.....w
  D5 CD 7E 9C 4E 62 20 0E D0 DB AD E7 7E 99 46 4F  ..~.Nb .....~.FO
  67 C7 A6 7E 2C 24 82 50 51 9F A7 B2 02 44 5B 30  g..~,$.PQ....D[0
  74 41 99 D9 83 69 EF AE 2E C0 FF C4 E6 4F F2 2F  tA...i.......O./
  95 FB 93 65 30 2A 2D 8D 20 88 83 B5 DE 35 B6 20  ...e0*-. ....5.
  47 17 30 25 60 FD E3 43 B9 D6 A4 F7 47 B6 6C 9F  G.0%`..C....G.l.
  47 FD 63 8E 7F A5 00 CE 6C 3E BC 95 23 69 ED D0  G.c.^?...l>..#i..
  69 4F BE 61 BD 30 C2 40 66 F6 F9 C3 3E C1 D7 8C  iO.a.0.@f...>...
  B0 C8 4A 2E 27 BE 13 6C 40 88 B0 13 A3 12 F4 50  ..J.'..l@.....P
  CA 92 D8 EB 4A E5 3F E2 64 A3 76 C7 5C 2B D8 89  ....J.?.d.v.\+..
  A3 6E C1 F7 0A C2 37 7A BD AF 14 4B 52 04 6B F2  .n....7z...KR.k.
  8F 4F C3 F8 00 90 BA 0F EC 6D B1 2D A8 18 0C A6  .O.......m.-....
  29 96 82 3B 5C BC D0 F4 2B BE 9C C5 8B 18 7A DE  )..;\...+.....z.
  C7 B5 10 2D 45 50 4F 77 ED F7 23 34 95 AF C3 2E  ...-EPOw..#4....
  B0 9B FA E9 DF                                    .....
...
```

Similar checks can be run on other low-level databases if you enable confidentiality by backend indexes as described below.

The settings for data confidentiality depend on the encryption capabilities of the JVM. For example, for details about the Sun/Oracle Java implementation, see the explanations in javax.crypto.Cipher. You can accept the default settings, or choose to specify the following:

• The cipher algorithm defining how the cleartext is encrypted and decrypted.

• The cipher mode of operation defining how a block cipher algorithm should transform data larger than a single block.

• The cipher padding defining how to pad the cleartext to reach appropriate size for the algorithm.

• The cipher key length, where longer key lengths strengthen encryption at the cost of more performance impact.

The default settings for confidentiality are `cipher-transformation: AES/CBC/PKCS5Padding` and `cipher-key-length: 128`. This means the algorithm is the Advanced Encryption Standard (AES), the cipher mode is Cipher Block Chaining (CBC), and the padding is PKCS#5 padding as described in RFC 2898:

PKCS #5: Password-Based Cryptography Specification. The syntax for the `cipher-transformation` is *algorithm/mode/padding*, and all three must be specified. When the algorithm does not require a mode, use `NONE`. When the algorithm does not require padding, use `NoPadding`. Use of larger `cipher-key-length` values can require that you install JCE policy files such as those for unlimited strength, as described in "Using Unlimited Strength Cryptography" in the *Security Guide*.

OpenDJ servers encrypt data using a symmetric key that is stored with the server configuration. The symmetric key is encrypted in turn with the server's public key that is also stored with the server configuration. When multiple servers are configured to replicate data as described in "Configuring Replication Settings", the servers replicate the keys as well, allowing any server replica to decrypt the data.

In addition to entry encryption, you can enable confidentiality by backend index, as long as confidentiality is enabled for the backend itself. Confidentiality hashes keys for equality type indexes using SHA-1, and encrypts the list of entries matching a substring key for substring indexes. The following example shows how to enable confidentiality for the `mail` index:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name mail \
 --set confidentiality-enabled:true \
 --no-prompt \
 --trustAll
```

After changing the index configuration, you can rebuild the index to enforce confidentiality immediately. For details, see "Configuring and Rebuilding Indexes".

Avoid using sensitive attributes in VLV indexes. Confidentiality cannot be enabled for VLV indexes.

Encrypting and decrypting data comes with costs in terms of cryptographic processing that reduces throughput and of extra space for larger encrypted values. In general, tests with default settings show that the cost of enabling confidentiality can be quite modest, but your results can vary based on your systems and on the settings used for `cipher-transformation` and `cipher-key-length`. Make sure you test your deployment to qualify the impact of confidentiality before enabling it in production.

4.8. Setting Disk Space Thresholds For Database Backends

Directory data growth depends on applications that use the directory. As a result, when directory applications add more data than they delete, the database backend grows until it fills the available disk space. The system can end up in an unrecoverable state if no disk space is available.

Database backends therefore have advanced properties, `disk-low-threshold` and `disk-full-threshold`. When available disk space falls below `disk-low-threshold`, the OpenDJ directory server only allows

updates from users and applications that have the `bypass-lockdown` privilege, as described in "About Privileges". When available space falls below `disk-full-threshold`, the OpenDJ directory server stops allowing updates, instead returning an `UNWILLING_TO_PERFORM` error to each update request.

An OpenDJ directory server continues to apply replication updates without regard to the thresholds. The server can therefore fill available disk space despite the thresholds, by accepting replication updates made on other servers. You can give yourself more time to react to the situation both by monitoring directory data growth and also by increasing the thresholds.

If growth across the directory service tends to happen quickly, set the thresholds higher than the defaults to allow more time to react when growth threatens to fill the disk. The following example sets `disk-low-threshold` to 2 GB `disk-full-threshold` to 1 GB for the `userRoot` backend:

```
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set "disk-low-threshold:2 GB" \
 --set "disk-full-threshold:1 GB" \
 --trustAll \
 --no-prompt
```

The properties `disk-low-threshold` and `disk-full-threshold` are listed as *advanced* properties. To examine their values with the **dsconfig** command, use the `--advanced` option as shown in the following example:

```
$ dsconfig \
 get-backend-prop \
 --advanced \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --property disk-low-threshold \
 --property disk-full-threshold \
 --trustAll \
 --no-prompt
Property            : Value(s)
--------------------:---------
disk-full-threshold : 1 gb
disk-low-threshold  : 2 gb
```

4.9. Updating an Existing Backend to Add a New Base DN

In addition to letting you create new backends as described in "Creating a New Database Backend", OpenDJ lets you add a new base DN to an existing backend.

The following example adds the suffix `o=example` to the existing backend `userRoot`:

```
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --add base-dn:o=example \
 --trustAll \
 --no-prompt
$ dsconfig \
 get-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --property base-dn \
 --trustAll \
 --no-prompt
Property : Value(s)
---------:-------------------------------
base-dn  : "dc=example,dc=com", o=example
```

Alternatively, you can update an existing backend in OpenDJ control panel (Directory Data > New Base DN, then select the existing backend from the dropdown Backend list, and enter the new Base DN name).

4.10. Deleting a Database Backend

You delete a database backend by using the **dsconfig delete-backend** command, described in the *Server Configuration Reference*.

When you delete a database backend by using the **dsconfig delete-backend** command, OpenDJ does not actually remove the database files for two reasons. First, a mistake could potentially cause lots of data to be lost. Second, deleting a large database backend could cause severe service degradation due to a sudden increase in I/O load.

Instead, after you run the **dsconfig delete-backend** command you must also manually remove the database backend files.

If you do run the **dsconfig delete-backend** command by mistake and have not yet deleted the actual files, then you can recover from the mistake by creating the backend again, reconfiguring the indexes that were removed, and rebuilding the indexes as described in "Configuring and Rebuilding Indexes".

Chapter 5
Configuring Connection Handlers

This chapter shows you how to configure OpenDJ servers to listen for directory client requests using connection handlers. You can view information about connection handlers in the OpenDJ control panel, and update the configuration using the **dsconfig** command, described in dsconfig(1) in the *Reference*.

In this chapter you will learn to:

• Enable client applications to access the directory over LDAP and secure LDAP (LDAPS)

• Enable client applications to access the directory over HTTP whether using DSML, or the REST style

• Enable monitoring using Java Management Extensions (JMX), or over Simple Network Management Protocol (SNMP)

• Enable automated processing of LDIF files

• Configure restrictions for client access such as requiring authentication or limiting the maximum number of concurrent connections

• Configure transport layer security for all relevant protocols

5.1. LDAP Client Access

You configure LDAP client access using the **dsconfig** command.

The reserved port number for LDAP is 389. Most examples in the documentation use 1389, which is accessible to non-privileged users.

To Change the LDAP Port Number

1. Change the port number using the **dsconfig** command.

   The following example changes the LDAP port number to 11389:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set listen-port:11389 \
 --trustAll \
 --no-prompt
```

2. Restart the connection handler so the change takes effect.

    To restart the connection handler, you disable it, then enable it again:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set enabled:false \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

5.2. Preparing For Secure Communications

This section shows how to prepare the server to use a file-based keystore to manage the keys essential to secure communications. For more information about the keys, see "About Certificates, Private Keys, and Secret Keys" in the *Security Guide*.

5.2.1. Importing Certificates

A client that sets up a secure connection with a server must be able to trust the server certificate. A server that uses mutual authentication (checking the client certificate) must be able to trust the client certificate. In either case, this involves finding the signing certificate in a keystore or a truststore.

In some cases, an OpenDJ server acts as a client of external services. For example, REST to LDAP can resolve OAuth 2.0 tokens by sending secure requests to an authorization server. The server can also connect to another LDAP server when using pass-through authentication.

The default Java truststore contains signing certificates from well-known CAs. If the CA certificate is not in the default truststore, or the certificate is self-signed, then you can import it into the server or client keystore as described here.

This section includes the following procedures:

- "To Import a Trusted Client Certificate"

- "To Import the Server Certificate"

- "To Import a Trusted CA Certificate"

To Import a Trusted Client Certificate

The following steps demonstrate using the **keytool** command to add a client application's binary format, self-signed certificate to the OpenDJ server keystore. This enables the OpenDJ server to recognize the self-signed client application certificate.

1. Import the client certificate:

    ```
    $ keytool \
     -import \
     -alias myapp-cert \
     -file myapp-cert.pem \
     -keystore /path/to/opendj/config/keystore \
     -storepass:file /path/to/opendj/config/keystore.pin \
     -storetype PKCS12 \
     -noprompt
    Certificate was added to keystore
    ```

2. Restart the server.

    This forces the server to read the updated truststore file:

    ```
    $ stop-ds --restart --quiet
    ```

To Import the Server Certificate

If your client uses a Java truststore, import the OpenDJ server signing certificate using the **keytool** command.

1. Export the server certificate from the server keystore.

    The following example shows the **keytool** command to export the self-signed server certificate in binary format. Notice that the keystore PIN is stored in a file, which is the default setting:

```
$ keytool \
 -export \
 -rfc \
 -alias server-cert \
 -file server-cert.pem \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12
Certificate stored in file <server-cert.pem>
```

2. Import the server certificate into the client truststore:

```
$ keytool \
 -import \
 -trustcacerts \
 -alias server-cert \
 -file server-cert.pem \
 -keystore my-keystore \
 -storetype PKCS12 \
 -storepass changeit \
 -noprompt
Certificate was added to keystore
```

To Import a Trusted CA Certificate

If you use self-signed certificates, or if your CA is not well-known, you can nevertheless import
the certificate as a CA certificate into a Java file-based truststore using the **keytool** command. The
application that relies on the truststore can then trust the certificate in the same way it trusts well-
known CA certificates.

*   Import the certificate as a CA certificate:

```
$ keytool \
 -import \
 -trustcacerts \
 -alias ca-cert \
 -file ca.crt \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -noprompt
Certificate was added to keystore
```

5.2.2. Setting Up Server Certificates

When you install the OpenDJ server, you can choose to configure secure connections and either
generate a key pair with a self-signed certificate, or import your own keystore. The default PKCS#12

keystore is `/path/to/opendj/config/keystore`, and the self-signed public key certificate has the alias `server -cert`. The password for the keystore and the private key is stored in cleartext in the file `/path/to/ opendj/config/keystore.pin`.

If you chose to set up a secure connection as part of the installation process, you can skip this section.

This section includes the following procedures:

- "To Set Up a CA-Signed Certificate"

- "To Set Up a Self-Signed Certificate"

- "To Use an Alternative Keystore Implementation"

To Set Up a CA-Signed Certificate

This procedure shows how to prepare a new key pair with a CA-signed certificate for use in setting up secure connections.

The high-level steps to perform are the following:

- Generate a server private key and public key certificate in your keystore.

- Issue a signing request to the CA, who responds with a CA-signed certificate.

- Import the CA-signed certificate where appropriate.

- Set up a key manager provider to use the keystore.

A detailed example follows:

1. Prepare the password for the keystore.

   By default, an OpenDJ server is configured to hold the password in a file, `/path/to/opendj/config/ keystore.pin`:

   ```
   $ touch /path/to/opendj/config/keystore.pin
   $ chmod 600 /path/to/opendj/config/keystore.pin
   # Add keystore and private key password in cleartext on a single line:
   $ vi /path/to/opendj/config/keystore.pin
   ```

2. Generate the server key pair (private key and public key certificate) using the Java **keytool** command.

   One step in verifying the certificate's validity is checking that the subject's FQDN matches the FQDN obtained from DNS.

The FQDN for the OpenDJ server, visible in the **status** command output, and under Server Details in the OpenDJ control panel, is set both as a `DNSName` in the certificate's `SubjectAlternativeName` list, and also in the CN of the certificate's subject name DN for backwards compatibility:

```
$ keytool \
 -genkeypair \
 -alias server-cert \
 -ext "san=dns:opendj.example.com" \
 -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -keypass:file /path/to/opendj/config/keystore.pin
```

> **Note**
>
> Notice that the `-storepass` and `-keypass` options take identical password arguments. An OpenDJ server uses the same password to protect the keystore and the private key.

If the server can respond on multiple FQDNs, then specify multiple subject alternative names when using the **keytool** command's `-ext` option. In the following example the primary FQDN is `opendj.example.com` and the alternative is `ldap.example.com`:

```
$ keytool \
 -genkeypair \
 -alias server-cert \
 -ext "san=dns:opendj.example.com,dns:ldap.example.com" \
 -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -keypass:file /path/to/opendj/config/keystore.pin
```

For an example showing how to use a wildcard certificate, see "To Set Up a Key Pair With a Wildcard Certificate".

3. Create a certificate signing request file for the generated certificate:

```
$ keytool \
 -certreq \
 -alias server-cert \
 -file server-cert.csr \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin
```

4. Have the CA sign the request (`server-cert.csr`).

See the instructions from your CA on how to provide the request.

The CA returns the signed certificate.

5. (Optional) If you have set up your own CA and signed the certificate, or are using a CA whose signing certificate is not included in the Java runtime environment, import the CA certificate into the keystore so that it can be trusted.

    Otherwise, when you import the signed certificate from an (unknown) CA, the **keytool** command fails to import the signed certificate with the message `keytool error: java.lang.Exception: Failed to establish chain from reply`.

    For an example command, see "To Import a Trusted CA Certificate".

6. Import the signed certificate from the CA reply into the keystore where you generated the server certificate.

    In this example the certificate from the reply is `server-cert.crt`:

```
$ keytool \
 -import \
 -alias server-cert \
 -file server-cert.crt \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -keypass:file /path/to/opendj/config/keystore.pin \
 -noprompt
```

7. Configure the file-based key manager provider for the keystore that you set up with the **keytool** command:

```
$ dsconfig \
 set-key-manager-provider-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Default Key Manager" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

    If you stored the keystore password somewhere besides the file, `/path/to/opendj/config/keystore.pin`, shown in the examples in this procedure, also adjust the `key-store-*` settings accordingly.

    At this point, the OpenDJ server can use the CA-signed certificate.

8. If you use a CA certificate that is not known to clients, such as a CA that you set up yourself rather than a well-known CA, import the CA certificate into the client application truststore. For an example command, see "To Import a Trusted CA Certificate".

   Otherwise the client application cannot trust the signature on the server certificate.

To Set Up a Self-Signed Certificate

This procedure shows how to prepare a new key pair with a self-signed certificate for use in setting up secure connections.

The high-level steps to perform are the following:

- Generate a server private key and public key certificate in your keystore.

- Self-sign the certificate.

- Set up a key manager provider to use the keystore.

To *replace the existing server key pair with a self-signed certificate and new private key*, first, use **keytool -delete -alias server-cert** to delete the existing keys, then generate a new key pair with the same alias. Either reuse the existing password in `keystore.pin`, or use a new password as shown in the steps below.

1. Prepare the password for the keystore.

   By default, an OpenDJ server is configured to hold the password in a file, `/path/to/opendj/config/keystore.pin`:

   ```
   $ touch /path/to/opendj/config/keystore.pin
   $ chmod 600 /path/to/opendj/config/keystore.pin
   # Add keystore and private key password in cleartext on a single line:
   $ vi /path/to/opendj/config/keystore.pin
   ```

2. Generate the key pair using the Java **keytool** command:

   ```
   $ keytool \
    -genkeypair \
    -alias server-cert \
    -ext "san=dns:opendj.example.com" \
    -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
    -keystore /path/to/opendj/config/keystore \
    -storetype PKCS12 \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -keypass:file /path/to/opendj/config/keystore.pin
   ```

   In this example, the OpenDJ server runs on a system with FQDN `opendj.example.com`. The keystore is created in the OpenDJ `config` directory.

> **Note**
>
> Notice that the `-storepass` and `-keypass` options take identical password arguments. An OpenDJ server uses the same password to protect the keystore and the private key.

If the server can respond on multiple FQDNs, then specify multiple subject alternative names when using the **keytool** command's `-ext` option. In the following example the primary FQDN is `opendj.example.com` and the alternative is `ldap.example.com`:

```
$ keytool \
 -genkeypair \
 -alias server-cert \
 -ext "san=dns:opendj.example.com,dns:ldap.example.com" \
 -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -keypass:file /path/to/opendj/config/keystore.pin
```

For an example showing how to use a wildcard certificate, see "To Set Up a Key Pair With a Wildcard Certificate".

3. Self-sign the server certificate:

```
$ keytool \
 -selfcert \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin
```

4. Configure the file-based key manager provider to access the keystore with keystore/private key password.

In this example, the alias is `server-cert` and the password is in the file, `/path/to/opendj/config/keystore.pin`.

If you are replacing a key pair with a self-signed certificate, reusing the `server-cert` alias and password stored in `keystore.pin`, then you can skip this step:

```
$ dsconfig \
 set-key-manager-provider-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Default Key Manager" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

If you stored the keystore password somewhere besides the file, `/path/to/opendj/config/keystore.pin`, shown in the examples in this procedure, also adjust the `key-store-*` settings accordingly.

At this point, the OpenDJ server can use the self-signed certificate, for example, for StartTLS and LDAPS or HTTPS connection handlers.

To Use an Alternative Keystore Implementation

To use an alternative keystore implementation, start with a different keystore type when generating the keypair:

1. Use one of the keystore types supported by the Java runtime environment:

    Java Keystore

    The basic Java keystore type is `JKS`:

    ```
    $ keytool \
     -genkeypair \
     -alias server-cert \
     -ext "san=dns:opendj.example.com" \
     -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
     -keystore /path/to/opendj/config/keystore.jks \
     -storetype JKS \
     -storepass:file /path/to/opendj/config/keystore.pin \
     -keypass:file /path/to/opendj/config/keystore.pin
    ```

    This is the keystore type if you do not specify a `-storetype` option.

    Java Cryptography Extension Keystore

    The `JCEKS` type lets you take advantage of additional Java cryptography extensions and stronger protection for private keys:

```
$ keytool \
-genkeypair \
-alias server-cert \
-ext "san=dns:opendj.example.com" \
-dname "CN=opendj.example.com,O=Example Corp,C=FR" \
-keystore /path/to/opendj/config/keystore.jceks \
-storetype JCEKS \
-storepass:file /path/to/opendj/config/keystore.pin \
-keypass:file /path/to/opendj/config/keystore.pin
```

LDAP Keystore

OpenDJ directory servers implement an `OpenDJ` security provider for LDAP and LDIF-based keystore services. Its keystore type is `LDAP`.

For details, see "Using an LDAP Keystore" in the *Security Guide*.

PKCS#11 device

A PKCS#11 device, such as an HSM, can be used as a keystore.

For details, see "Using a Hardware Security Module" in the *Security Guide*.

PKCS#12 Keystore

The `PKCS12` type lets you use a PKCS#12 format file. This is the default for OpenDJ servers. It is a standard format and is interoperable with other systems that do not necessarily depend on a Java runtime environment:

```
$ keytool \
-genkeypair \
-alias server-cert \
-ext "san=dns:opendj.example.com" \
-dname "CN=opendj.example.com,O=Example Corp,C=FR" \
-keystore /path/to/opendj/config/keystore \
-storetype PKCS12 \
-storepass:file /path/to/opendj/config/keystore.pin \
-keypass:file /path/to/opendj/config/keystore.pin
```

2. After using an alternate keystore type, make sure that you set up:

   • The key manager provider to open the correct keystore with the correct credentials.

     Any components using that key manager provider to use the correct certificate alias.

5.2.3. Working With Test Key Pairs

This section includes procedures for working with test key pairs. The procedures use the **openssl** command to perform actions that are not necessarily possible with the Java **keytool** command. OpenSSL software is available from https://www.openssl.org/.

> **Important**
>
> OpenSSL software is independent from and not associated with the OpenDJ project. The examples shown in this section might not work with your version of OpenSSL. See the **openssl** documentation for your version.
>
> Examples in this section were originally written with OpenSSL version 1.0.2.

This section includes the following procedures:

- "To Set Up a CA Signing Certificate"

- "To Sign a Certificate Signing Request"

- "To Set Up a Key Pair With a Wildcard Certificate"

To Set Up a CA Signing Certificate

Follow these steps to set up a CA key pair with a signing certificate:

1. Generate a private key for the CA:

   ```
   $ openssl \
    genpkey \
    -algorithm RSA \
    -out ca.key \
    -pkeyopt rsa_keygen_bits:4096
   ```

2. Self-sign a corresponding certificate:

   ```
   $ openssl \
    req \
    -new \
    -x509 \
    -days 7300 \
    -subj "/C=FR/O=Example Corp/CN=example.com" \
    -key ca.key \
    -out ca.crt
   ```

   The signing certificate is `ca.crt`, and it corresponds to the private key `ca.key`.

To Sign a Certificate Signing Request

Follow these steps to use a CA signing certificate to sign a certificate signing request (CSR):

1. If you have not already done so, set up a CA key pair as described in "To Set Up a CA Signing Certificate".

2. Obtain a CSR file for the certificate to sign.

3. Create a signed certificate from the CSR.

   The following example preserves the SAN for a server certificate generated using the **keytool** command and the `-ext "san=dns:opendj.example.com"` option:

   ```
   $ openssl \
    x509 \
    -req \
    -in server-cert.csr \
    -CA ca.crt \
    -CAkey ca.key \
    -CAcreateserial \
    -extfile \
     <(cat /etc/ssl/openssl.cnf \
     <(printf "[SAN]\nsubjectAltName=DNS:opendj.example.com")) \
    -extensions SAN \
    -out server-cert.crt
   ```

   The following example preserves the SAN for a wildcard certificate as described in "To Set Up a Key Pair With a Wildcard Certificate":

   ```
   $ openssl \
    x509 \
    -req \
    -in example.com.csr \
    -CA ca.crt \
    -CAkey ca.key \
    -CAcreateserial \
    -extfile \
     <(cat /etc/ssl/openssl.cnf \
     <(printf "[SAN]\nsubjectAltName=DNS:*.example.com")) \
    -extensions SAN \
    -out example.com.crt
   ```

   In both examples, the certificates to return to the requestors are the `.crt` files.

To Set Up a Key Pair With a Wildcard Certificate

A wildcard certificate uses a `*` to replace the top-level subdomain in the subject FQDN, and can list domains in the subject alternative domain list.

The FQDN for an OpenDJ server, visible in the **status** command output, and under Server Details in the OpenDJ control panel, must also match a `DNSName` value or pattern in the certificate's `SubjectAlternativeName` list.

Follow these steps to set up a key pair with a wildcard certificate:

1. Generate a private key:

```
$ openssl \
 genpkey \
 -algorithm RSA \
 -pkeyopt rsa_keygen_bits:2048 \
 -out example.com.key
```

2. Generate a certificate signing request for the CA:

```
$ openssl \
 req \
 -new \
 -sha256 \
 -key example.com.key \
 -subj "/C=FR/O=Example Corp/CN=*.example.com" \
 -reqexts SAN \
 -config \
  <(cat /etc/ssl/openssl.cnf \
  <(printf "[SAN]\nsubjectAltName=DNS:*.example.com")) \
 -out example.com.csr
```

3. Get the CA to sign the request and return the certificate.

   For an example of how to do this yourself, see "To Sign a Certificate Signing Request".

4. (Optional) Convert the results into a PKCS#12 format Java keystore file.

```
$ openssl \
 pkcs12 \
 -export \
 -in example.com.crt \
 -inkey example.com.key \
 -name server-cert \
 -CAfile ca.crt \
 -password file:/path/to/opendj/config/keystore.pin \
 -out /path/to/opendj/config/keystore
```

This keystore can be used for testing the wildcard certificate with an OpenDJ server as in the following example:

```
$ ldapsearch \
 --port 1636 \
 --hostname opendj.example.com \
 --baseDN dc=example,dc=com \
 --useSSL \
 "(uid=bjensen)" \
 cn
The server is using the following certificate:
    Subject DN:  CN=*.example.com, O=Example Corp, C=FR
    Issuer DN:   CN=example.com, O=Example Corp, C=FR
    Validity:  <validity-dates>
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

5.3. LDAP Client Access With Transport Layer Security

StartTLS negotiations start on the unsecure LDAP port, and then protect communication with the client. You can configure StartTLS when setting up an OpenDJ server, or later using the **dsconfig** command.

To Enable StartTLS on the LDAP Port

1. Make sure you have a server certificate installed:

```
$ keytool \
 -list \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12
...
server-cert, <date>, PrivateKeyEntry
...
```

2. Activate StartTLS on the current LDAP port:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set allow-start-tls:true \
 --set key-manager-provider:"Default Key Manager" \
 --set trust-manager-provider:"JVM Trust Manager" \
 --trustAll \
 --no-prompt
```

The change takes effect. No need to restart the server.

5.4. LDAP Client Access Over SSL

An LDAP connection handler configured to use LDAPS (LDAP/SSL) allows only secure connections from client applications. You can configure LDAPS when setting up an OpenDJ server, or later using the **dsconfig** command.

The reserved port number for LDAPS is 636. Most examples in the documentation use 1636, which is accessible to non-privileged users.

To Set Up LDAPS Access

If LDAPS access was not configured at during setup, follow these steps:

1. Make sure a server certificate and associated private key are available:

    ```
    $ keytool \
     -list \
     -alias server-cert \
     -keystore /path/to/opendj/config/keystore \
     -storepass:file /path/to/opendj/config/keystore.pin \
     -storetype PKCS12
    server-cert, <date>, PrivateKeyEntry
    ```

2. Configure the server to activate LDAPS access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set enabled:true \
 --set listen-port:1636 \
 --set use-ssl:true \
 --trustAll \
 --no-prompt
```

3. (Optional) If the deployment requires SSL client authentication, set the properties `ssl-client-auth-policy` and `trust-manager-provider` appropriately.

To Change the LDAPS Port Number

1. Change the port number using the **dsconfig** command.

   The following example changes the LDAPS port number to 11636:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set listen-port:11636 \
 --trustAll \
 --no-prompt
```

2. Restart the connection handler so the change takes effect.

   To restart the connection handler, you disable it, then enable it again:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set enabled:false \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

5.5. Restricting Client Access

Using an OpenDJ server's global configuration properties, you can add global restrictions on how clients access the server. These settings are per server, and so must be set independently on each server.

These global settings are fairly coarse-grained. For a full discussion of the rich set of administrative privileges and fine-grained access control instructions that OpenDJ servers support, see "*Configuring Privileges and Access Control*".

Consider the following global configuration settings:

bind-with-dn-requires-password

Whether the server should reject any simple bind request that contains a DN but no password. Default: `true`

To change this setting use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set bind-with-dn-requires-password:false \
 --no-prompt \
 --trustAll
```

max-allowed-client-connections

Restricts the number of concurrent client connections to this server. Default: 0, meaning no limit is set.

To set a limit of 64K use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set max-allowed-client-connections:65536 \
 --no-prompt \
 --trustAll
```

reject-unauthenticated-requests

Rejects any request (other than bind or StartTLS requests) received from a client that has not yet been authenticated, whose last authentication attempt was unsuccessful, or whose last authentication attempt used anonymous authentication. Default: false.

To shut down anonymous binds use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set reject-unauthenticated-requests:true \
 --no-prompt \
 --trustAll
```

return-bind-error-messages

Does not restrict access, but by default prevents an OpenDJ server from returning extra information about why a bind failed, as that information could be used by an attacker. Instead, the information is written to the server errors log. Default: false.

To have the server return additional information about why a bind failed use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set return-bind-error-messages:true \
 --no-prompt \
 --trustAll
```

5.6. TLS Protocols and Cipher Suites

When a server and client negotiate a secure connection, they negotiate use of a common protocol and cipher suite. If they cannot negotiate a common protocol and cipher suite, they will fail to set up a secure connection.

Furthermore, researchers continue to find vulnerabilities in protocols and cipher suites. If the server continues to support those protocols and cipher suites, then clients can use them when negotiating connections. Attackers can then exploit the vulnerabilities. It is therefore important to understand how to read and to restrict the list of supported protocols and cipher suites.

OpenDJ servers depend on the underlying JVM to support security protocols and cipher suites. By default, all supported protocols and cipher suites are available. For details, see the documentation for the JVM. For Oracle Java, see the *Java Cryptography Architecture Oracle Providers Documentation* describing the The `SunJSSE` Provider.

Bear in mind that support for protocols and cipher suites can be added and removed in Java update releases. The protocols and cipher suites available are also determined by the security policy. For example, see "Using Unlimited Strength Cryptography" in the *Security Guide*.

This section includes the following procedures:

• "To List Protocols and Cipher Suites"

• "To Restrict Protocols and Cipher Suites"

• "To Restrict Protocols For Command-Line Tools"

To List Protocols and Cipher Suites

•   To list the protocols and cipher suites that an OpenDJ server supports, read the `supportedTLSProtocols` and `supportedTLSCiphers` attributes of the root DSE:

```
$ ldapsearch \
 --port 1389 \
 --baseDN "" \
 --searchScope base \
 "(objectclass=*)" \
 supportedTLSCiphers supportedTLSProtocols
```

A `supportedTLSCiphers` name, such as `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`, identifies the key attributes of the cipher suite:

TLS

> Specifies the protocol, in this case TLS.

ECDHE_RSA

> Specifies the key exchange algorithm used to determine how the client and server authenticate during the handshake phase.

> In this example, an elliptic curve variant of the Diffie-Hellman key exchange is used, where a random value chosen by the client is encrypted with the server's RSA public key.

WITH_AES_256_GCM

> Specifies the bulk encryption algorithm, including the key size or initialization vectors.

> This example specifies the Advanced Encryption Standard (AES) with 256-bit key size and Galois/Counter Mode (GCM) block cipher mode.

SHA384

> Specifies the message authentication code algorithm used to create the message digest, which is a cryptographic hash of each block in the message stream.

> In this example, the SHA-2 hash function, SHA-384, is used.

A `supportedTLSProtocols` name identifies the protocol and version, such as `TLSv1.2`.

To Restrict Protocols and Cipher Suites

Choosing which protocol versions and cipher suites to allow for negotiating secure connections depends on known vulnerabilities, estimations of what is secure, and what peers support.

The default support is backward-compatible with old clients, meaning that if you do nothing it is possible for a client to use a protocol version that is known to have vulnerabilities, or to negotiate an insecure or very weakly secure connection. To avoid this, limit the protocols and cipher suites that you allow.

You can limit supported protocols and cipher suites by setting the properties, `ssl-protocol` and `ssl-cipher-suite`. These are available on connection handlers, and on components that connect to remote services including the global Crypto Manager component used for replication.

This procedure is based on server-side TLS recommendations from the Mozilla Operations Security team taken at the time of this writing. Recommendations evolve. Make sure you use current recommendations when configuring security settings.

The examples in this procedure assume you have installed an unlimited encryption strength policy:

1. For each cipher suite key algorithm to support, create a key pair using the supported key algorithm.

   The following example adds two key pairs to the default PKCS#12 keystore, with one key using RSA and the other key using elliptic curve.

   ```
   $ keytool \
    -genkeypair \
    -alias server-cert-rsa \
    -keyalg RSA \
    -ext "san=dns:opendj.example.com" \
    -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
    -keystore /path/to/opendj/config/keystore \
    -storetype PKCS12 \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -keypass:file /path/to/opendj/config/keystore.pin
   $ keytool \
    -genkeypair \
    -alias server-cert-ec \
    -keyalg EC \
    -ext "san=dns:opendj.example.com" \
    -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
    -keystore /path/to/opendj/config/keystore \
    -storetype PKCS12 \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -keypass:file /path/to/opendj/config/keystore.pin
   ```

2. On the components you use, explicitly set the supported protocols and cipher suites.

   The following example adjusts settings for the LDAP and LDAPS connection handlers:

   ```
   $ dsconfig \
    set-connection-handler-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --handler-name "LDAP Connection Handler" \
    --add ssl-protocol:TLSv1.2 \
    --add ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
    --add ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
    --add ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
    --add ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
   ```

```
--add ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV \
--no-prompt \
--trustAll
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --add ssl-protocol:TLSv1.2 \
 --add ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
 --add ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
 --add ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
 --add ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
 --add ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV \
 --no-prompt \
 --trustAll
```

The `TLS_ECDHE_EC*` cipher suites call for a server certificate with an elliptic curve key algorithm. The `TLS_ECDHE_RSA*` cipher suites call for a server certificate with an RSA key algorithm.

The `TLS_EMPTY_RENEGOTIATION_INFO_SCSV` cipher suite is a renegotiation information extension with a special Signaling Cipher Suite Value (SCSV) to help older clients properly complete a handshake.

3. On the components you use, set the multivalued property, `ssl-cert-nickname`, to identify each certificate alias:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set enabled:true \
 --set listen-port:1389 \
 --set allow-start-tls:true \
 --set ssl-cert-nickname:server-cert-ec \
 --set ssl-cert-nickname:server-cert-rsa \
 --set key-manager-provider:"Default Key Manager" \
 --set trust-manager-provider:"JVM Trust Manager" \
 --trustAll \
 --no-prompt
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set listen-port:1636 \
 --set enabled:true \
 --set use-ssl:true \
 --set ssl-cert-nickname:server-cert-ec \
```

```
--set ssl-cert-nickname:server-cert-rsa \
--trustAll \
--no-prompt
```

When negotiating a secure connection, the server can now use either key.

To Restrict Protocols For Command-Line Tools

You can specify which protocol versions to allow when command-line tools negotiate secure connections with LDAP servers.

The command-line tools depend on a system property, `org.opends.ldaps.protocols`. This property takes a comma-separated list of protocols. The default is constructed from the list of all protocols the JVM supports, removing protocol names starting with `SSL`. For example, if support is enabled in the JVM for versions 1, 1.1., and 1.2 of the TLS protocol, then the default is `"TLSv1,TLSv1.1,TLSv1.2"`.

- Restrict the protocols to use by setting the property in one of the following ways:

  - Set the property by editing the `java-args` for the command in `config/java.properties`.

    For example, to restrict the protocol to TLS v1.2 when the **status** command negotiates a secure administrative connection, edit the corresponding line in `config/java.properties`:

    ```
    status.java-args=-Xms8m -client -Dorg.opends.ldaps.protocols=TLSv1.2
    ```

  - Set the property at runtime when running the command.

    The following example restricts the protocol to TLS v1.2 when the **status** command negotiates a secure administrative connection:

    ```
    $ export OPENDJ_JAVA_ARGS="-Dorg.opends.ldaps.protocols=TLSv1.2"
    $ status \
     --bindDN "cn=Directory Manager" \
     --bindPassword password \
     --trustAll
    ```

5.7. Client Certificate Validation and the Directory

This section clarifies the roles that client applications' X.509 digital certificates play in establishing secure connections and in authenticating the client as a directory user. Be aware that establishing a secure connection happens before the server handles the LDAP or HTTP requests that the client sends over the secure connection. Establishing a secure connection is handled separately from authenticating a client as a directory user, even though both processes can involve the client's certificate.

When a client and a server negotiate a secure connection over LDAPS or HTTPS, or over LDAP using the StartTLS operation, they can use public key cryptography to authenticate each other.

The server, client, or both present certificates to each other. By default, OpenDJ LDAPS and HTTPS connection handlers are configured to present the server certificate, and to consider the client certificate optional. The connection handler property `ssl-client-auth-policy` makes the latter behavior configurable. For the DSML and REST to LDAP gateways, HTTPS negotiation is handled by the web application container where the gateway runs. See the web application container documentation for details on configuring how the container handles the client certificate.

One step toward establishing a secure connection involves validating the certificate that was presented by the other party. Part of this is trusting the certificate. The certificate identifies the client or server and the CA certificate used to sign the client or server certificate. The validating party checks that the other party corresponds to the one identified by the certificate, and checks that the signature can be trusted. If the signature is valid, and the CA certificate used to sign the certificate can be trusted, then the certificate can be trusted. This part of the validation process is also described briefly in "How Keys are Used" in the *Security Guide*.

Certificates can be revoked after they are signed. Therefore, the validation process can involve checking whether the certificate is still valid. Two different methods for performing this validation use the Online Certificate Status Protocol (OCSP) or Certificate Revocation Lists (CRLs). OCSP is a newer solution that provides an online service to handle the revocation check for a specific certificate. CRLs are potentially large lists of user certificates that are no longer valid or that are on hold. A CRL is signed by the CA. The validating party obtains the CRL and checks that the certificate being validated is not listed. For a brief comparison, see *OCSP: Comparison to CRLs*. A certificate can include links to contact the OCSP responder or to the CRL distribution point. The validating party can use these links to check whether the the certificate is still valid.

In both cases, the CA who signed the certificate acts as the OCSP responder or publishes the CRLs. When establishing a secure connection with a client application, an OpenDJ server relies on the CA for OCSP and CRLs. This is the case even when OpenDJ is the repository for the CRLs.

An OpenDJ directory service is a logical repository for certificates and CRLs. For example, an OpenDJ directory server can store CRLs in a `certificateRevocationList` attribute as in the following example entry:

```
dn: cn=My CA,dc=example,dc=com
objectClass: top
objectClass: applicationProcess
objectClass: certificationAuthority
cn: My CA
authorityRevocationList;binary: Base64-encoded ARL
cACertificate;binary:: Base64-encoded CA certificate
certificateRevocationList;binary:: Base64-encoded CRL
```

The CRL could then be replicated to other OpenDJ directory servers for high availability. (Notice the ARL in this entry. An ARL is like a CRL, but for CA certificates.)

Again, despite being a repository for CRLs, an OpenDJ directory server does not use the CRLs directly when checking a client certificate. Instead, when negotiating a secure connection, the server depends on the JVM security configuration. The JVM configuration governs whether validation uses

OCSP, CRLs, or both. As described in the *Java PKI Programmer's Guide* under *Support for the CRL Distribution Points Extension*, and *Appendix C: On-Line Certificate Status Protocol (OCSP) Support*, the JVM relies on system properties that define whether to use the CRL distribution points defined in certificates, and how to handle OCSP requests. These system properties can be set system-wide in `$JAVA_HOME/lib/security/java.security` (`$JAVA_HOME/jre/lib/security/java.security` for the JDK). The JVM handles revocation checking without the OpenDJ server's involvement.

After a connection is negotiated, the server can authenticate a client application at the LDAP level based on the certificate. For details, see "Authenticating Client Applications With a Certificate" in the *Security Guide*.

OCSP and obtaining CRLs depend on network access to the CA. If OpenDJ servers or the DSML or REST to LDAP gateways run on a network where the CA is not accessible, and the deployment nevertheless requires OSCP or checking CRLs for client application certificates, then you must provide some alternative means to handle OCSP or CRL requests. The JVM can be configured to use a locally available OCSP responder, for example, and that OCSP responder might depend on an OpenDJ directory server. If the solution depends on CRLs, you could regularly update the CRLs in the directory with copies of the CA CRLs obtained by other means.

5.8. RESTful Client Access Over HTTP

OpenDJ software offers two ways to give RESTful client applications HTTP access to directory user data as JSON resources:

- Enable the listener on an OpenDJ server to respond to REST requests.

  With this approach, you do not need to install additional software.

  For details, see the following procedures:

  - "To Set Up HTTP Access"

  - "To Set Up HTTPS Access"

  - "To Set Up REST Access to User Data"

  - "To Set Up HTTP Authorization"

- Configure the external REST to LDAP gateway Servlet to access the directory service.

  With this approach, you must install the gateway separately.

  For details, see "To Set Up OpenDJ REST to LDAP Gateway".

OpenDJ servers also expose administrative data over HTTP. For details, see "To Set Up REST Access to Administrative Data".

The REST to LDAP mappings follow these rules to determine JSON property types:

- If the LDAP attribute is defined in the LDAP schema, then the REST to LDAP mapping uses the most appropriate type in JSON. For example, numbers appear as JSON numbers, and booleans as booleans.

- If the LDAP attribute only has one value, then it is returned as a scalar.

- If the LDAP attribute has multiple values, then the values are returned in an array.

To Set Up HTTP Access

OpenDJ servers have a handler for HTTP connections. This handler exposes directory data over HTTP, including the RESTful API demonstrated in "*Performing RESTful Operations*" in the *Developer's Guide*. The HTTP connection handler can be enabled during the setup process.

This procedure shows you how to enable the HTTP connection handler if you did not enable the connection handler during the setup process.

The HTTP connection handler exposes directory data at HTTP endpoints. After you set up the HTTP connection handler, make sure that at least one HTTP endpoint is enabled, for example by following the steps described in "To Set Up REST Access to User Data", or the steps described in "To Set Up REST Access to Administrative Data". It is possible to enable multiple HTTP endpoints, as long as their base paths are different.

1. If you did not enable the HTTP connection handler during the setup process with the `--httpPort` or `--httpsPort` option, create the connection handler configuration:

    ```
    $ dsconfig \
     create-connection-handler \
     --hostname opendj.example.com \
     --port 4444 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password \
     --handler-name "HTTP Connection Handler" \
     --type http \
     --set enabled:true \
     --set listen-port:8080 \
     --no-prompt \
     --trustAll
    ```

2. (Optional) Enable an HTTP access log.

    - The following command enables JSON-based HTTP access logging as described in "Configuring JSON Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

• The following command enables HTTP access logging as described in "Standard HTTP Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

3. (Optional) If necessary, change the connection handler configuration using the **dsconfig** command.

To Set Up HTTPS Access

The following steps demonstrate how to change the default HTTP connection handler configuration to use only HTTPS:

1. Make sure a server certificate and associated private key are available:

```
$ keytool \
 -list \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12
...
server-cert, <date>, PrivateKeyEntry
...
```

2. Disable the HTTP connection handler to prevent (cleartext) HTTP access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "HTTP Connection Handler" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

3. Configure the HTTP connection handler to use HTTPS access.

   The following example shows how to set the port to 8443 and perform TLS using the default server certificate:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "HTTP Connection Handler" \
 --set enabled:true \
 --set listen-port:8443 \
 --set use-ssl:true \
 --set key-manager-provider:"Default Key Manager" \
 --set trust-manager-provider:"JVM Trust Manager" \
 --trustAll \
 --no-prompt
```

4. (Optional) Enable the HTTP access log.

   • The following command enables JSON-based HTTP access logging as described in "Configuring JSON Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

   • The following command enables HTTP access logging as described in "Standard HTTP Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

5. (Optional) If the deployment requires SSL client authentication, set the properties `ssl-client-auth-policy` and `trust-manager-provider` appropriately.

6. After you set up the HTTP connection handler, make sure that at least one HTTP endpoint configuration object is enabled.

   For details, see "To Set Up REST Access to User Data" or "To Set Up REST Access to Administrative Data".

To Set Up REST Access to User Data

The way directory data appears to client applications is configurable. You can configure one or more Rest2ldap endpoints to expose user directory data over HTTP. The mapping defined for the Rest2ldap endpoint defines a mapping between JSON resources and LDAP entries. The mapping is expressed in a configuration file, by default `/path/to/opendj/config/rest2ldap/endpoints/api/example-v1.json`. The configuration is described in "*REST to LDAP Configuration*" in the *Reference*.

The default Rest2ldap endpoint exposes the RESTful API demonstrated in "*Performing RESTful Operations*" in the *Developer's Guide*. The default mapping works out of the box with Example.com data generated as part of the setup process and with example data imported from Example.ldif:

1. (Optional) If necessary, change the properties of the default Rest2ldap endpoint, or create a new endpoint.

   A Rest2ldap HTTP endpoint named `/api` after its `base-path` is provided by default. The `base-path` must be the same as the name, and is read-only after creation. By default, the `/api` endpoint requires authentication.

   The following example enables the `/api` endpoint using the default mapping and HTTP Basic authorization. Adjust these settings as necessary:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --endpoint-name /api \
 --set authorization-mechanism:"HTTP Basic" \
 --set config-directory:config/rest2ldap/endpoints/api \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

Alternatively, you can create another Rest2ldap endpoint to expose a different view of the directory data, or to publish data under an alternative base path, such as `/rest`:

```
$ dsconfig \
 create-http-endpoint \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --endpoint-name /rest \
 --type rest2ldap-endpoint \
 --set authorization-mechanism:"HTTP Basic" \
 --set config-directory:config/rest2ldap/endpoints/api \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

2. (Optional) If necessary, adjust the endpoint configuration to use an alternative HTTP authorization mechanism.

   By default, the Rest2ldap endpoint maps HTTP Basic authentication to LDAP authentication to set the authorization identity for operations. You can change the `authorization-mechanism` setting to use a different HTTP authorization mechanism as described in "To Set Up HTTP Authorization".

3. (Optional) Try reading a resource.

   The following example demonstrates reading the resource that corresponds to Barbara Jensen's entry as a JSON resource:

```
$ curl http://bjensen:hifalutin@opendj.example.com:8080/api/users/bjensen?_prettyPrint=true
{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : { },
  "userName" : "bjensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen",
  "manager" : {
    "_id" : "trigden",
    "displayName" : "Torrey Rigden"
  }
}
```

4. (Optional) If the HTTP connection handler is configured to use HTTPS, try reading an entry over HTTPS.

   The following example writes the (self-signed) server certificate into a trust store file, and uses the file to trust the server when setting up the HTTPS connection:

```
$ keytool \
 -export \
 -rfc \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -file server-cert.pem
Certificate stored in file <server-cert.pem>

$ curl \
 --cacert server-cert.pem \
 --user bjensen:hifalutin \
 https://opendj.example.com:8443/api/users/bjensen?_prettyPrint=true
{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : { },
  "userName" : "bjensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
```

```
      "givenName" : "Barbara",
      "familyName" : "Jensen"
    },
    "description" : "Original description",
    "contactInformation" : {
      "telephoneNumber" : "+1 408 555 1862",
      "emailAddress" : "bjensen@example.com"
    },
    "uidNumber" : 1076,
    "gidNumber" : 1000,
    "homeDirectory" : "/home/bjensen",
    "manager" : {
      "_id" : "trigden",
      "displayName" : "Torrey Rigden"
    }
}
```

Notice the `--cacert server-cert.pem` option used with the **curl** command. This is the way to specify a self-signed server certificate when using HTTPS.

To Set Up HTTP Authorization

HTTP authorization mechanisms define how OpenDJ servers authorize client HTTP requests to directory data. Authorization mechanisms map credentials from an HTTP-based protocol, such as HTTP Basic authentication or OAuth 2.0, to LDAP credentials.

Multiple HTTP authorization mechanisms can be enabled simultaneously, and assigned to HTTP endpoints, such as Rest2ldap endpoints described in "To Set Up REST Access to User Data" or the Admin endpoint described in "To Set Up REST Access to Administrative Data".

By default, these HTTP authorization mechanisms are supported:

HTTP Anonymous

> Handle anonymous HTTP requests, optionally binding with a specified DN.
>
> If no bind DN is specified (default), anonymous LDAP requests are used.
>
> This mechanism is enabled by default unless the server was installed with the **setup** command option, `--productionMode`.

HTTP Basic (enabled by default)

> Handle HTTP Basic authentication requests by mapping the HTTP Basic identity to a user's directory account for the underlying LDAP operation.
>
> By default, the Exact Match identity mapper with its default configuration is used to map the HTTP Basic user name to an LDAP `uid`. The OpenDJ server then searches in all public naming contexts to find the user's entry based in the `uid` value.

HTTP OAuth2 CTS

Handle OAuth 2.0 requests as an OAuth 2.0 resource server, where an OpenDJ service acts as an OpenAM Core Token Service (CTS) store.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the OpenDJ server tries to resolve the access token against the CTS data that it serves for OpenAM. If the access token resolves correctly (is found in the CTS data and has not expired), the OpenDJ server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present and the token is valid, it maps the user identity to a user's directory account for the underlying LDAP operation.

This mechanism makes it possible to resolve access tokens by making an internal request, avoiding a request to OpenAM. *This mechanism does not, however, ensure that the token requested will have already been replicated to the directory server where the request is routed.*

OpenAM's CTS store is constrained to a specific layout. The `authzid-json-pointer` must therefore use `userName/0` for the user identifier.

HTTP OAuth2 OpenAM

Handle OAuth 2.0 requests as an OAuth 2.0 resource server, where the OpenDJ server sends requests to OpenAM for access token resolution.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the OpenDJ server requests token information from OpenAM. If the access token is valid, the OpenDJ server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, it maps the user identity to a user's directory account for the underlying LDAP operation.

As access token resolution requests ought to be sent over HTTPS, you can configure a trust store manager if necessary to trust the authorization server certificate, and a key store manager to obtain the OpenDJ server certificate if the authorization server requires mutual authentication.

HTTP OAuth2 Token Introspection (RFC7662)

Handle OAuth 2.0 requests as an OAuth 2.0 resource server, where the OpenDJ server sends requests to an RFC 7662-compliant authorization server for access token resolution.

RFC 7662, *OAuth 2.0 Token Introspection*, defines a standard method for resolving access tokens. The OpenDJ server must be registered as a client of the authorization server.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the OpenDJ server requests token introspection from the authorization server. If the access token is valid, the OpenDJ server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, it maps the user identity to a user's directory account for the underlying LDAP operation.

As access token resolution requests ought to be sent over HTTPS, you can configure a trust store manager if necessary to trust the authorization server certificate, and a key store manager to obtain the OpenDJ server certificate if the authorization server requires mutual authentication.

> **Note**
>
> The HTTP OAuth2 File mechanism is an internal interface intended for testing and not supported for production use.

When more than one authentication mechanism is specified, mechanisms are applied in the following order:

- If the client request has an `Authorization` header, and an OAuth 2.0 mechanism is specified, the server attempts to apply the OAuth 2.0 mechanism.

- If the client request has an `Authorization` header, or has the custom credentials headers specified in the configuration, and an HTTP Basic mechanism is specified, the server attempts to apply the Basic Auth mechanism.

- Otherwise, if an HTTP anonymous mechanism is specified, and none of the previous mechanisms apply, the server attempts to apply the mechanism for anonymous HTTP requests.

There are many possibilities when configuring HTTP authorization mechanisms. *This procedure shows only one OAuth 2.0 example.*

The example that follows demonstrates a server configured for tests (insecure connections) to request OAuth 2.0 token information from OpenAM. It uses settings as listed in "Settings for OAuth 2.0 Example With OpenAM".

Download ForgeRock Access Management or OpenAM software from https://backstage.forgerock.com/downloads.

Settings for OAuth 2.0 Example With OpenAM

Setting	Value
OpenAM URL	`http://openam.example.com:8088/openam`
Authorization server endpoint	`/oauth2/tokeninfo` (top-level realm)
Identity repository	`opendj.example.com:1389` with `Example.ldif` data
OAuth 2.0 client ID	`myClientID`
OAuth 2.0 client secret	`password`
OAuth 2.0 client scopes	`read`, `uid`, `write`
Rest2ldap configuration	Default settings. See "To Set Up REST Access to User Data".

Read the ForgeRock Access Management documentation if necessary to install and configure OpenAM. Then follow these steps to try the demonstration:

1. Update the default HTTP OAuth2 OpenAM configuration:

```
$ dsconfig \
 set-http-authorization-mechanism-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --mechanism-name "HTTP OAuth2 OpenAM" \
 --set enabled:true \
 --set token-info-url:http://openam.example.com:8088/openam/oauth2/tokeninfo \
 --no-prompt \
 --trustAll
```

2. Update the default Rest2ldap endpoint configuration to use HTTP OAuth2 OpenAM as the authorization mechanism:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --endpoint-name "/api" \
 --set authorization-mechanism:"HTTP OAuth2 OpenAM" \
 --no-prompt \
 --trustAll
```

3. Obtain an access token with the appropriate scopes:

```
$ curl \
 --request POST \
 --user "myClientID:password" \
 --data "grant_type=password&username=bjensen&password=hifalutin&scope=read%20uid%20write" \
 http://openam.example.com:8088/openam/oauth2/access_token
{
 "access_token": "token-string",
 "scope": "uid read write",
 "token_type": "Bearer",
 "expires_in": 3599
}
```

In production systems, make sure you use HTTPS when obtaining access tokens.

4. Request a resource at the Rest2ldap endpoint using HTTP Bearer authentication with the access token:

```
$ curl \
 --header "Authorization: Bearer token-string" \
 http://opendj.example.com:8080/api/users/bjensen?_prettyPrint=true
{
  "_id": "bjensen",
  "_rev": "<revision>",
  "_schema": "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta": {},
  "userName": "bjensen@example.com",
  "displayName": ["Barbara Jensen", "Babs Jensen"],
  "name": {
    "givenName": "Barbara",
    "familyName": "Jensen"
  },
  "description": "Original description",
  "contactInformation": {
    "telephoneNumber": "+1 408 555 1862",
    "emailAddress": "bjensen@example.com"
  },
  "uidNumber": 1076,
  "gidNumber": 1000,
  "homeDirectory": "/home/bjensen",
  "manager": {
    "_id": "trigden",
    "displayName": "Torrey Rigden"
  }
}
```

In production systems, make sure you use HTTPS when presenting access tokens.

To Set Up REST Access to Administrative Data

By default, the HTTP connection handler exposes an Admin endpoint with base path `/admin` that
is protected by the HTTP Basic authorization mechanism. (This endpoint is not available through
the gateway.) The APIs for configuration and monitoring OpenDJ servers are under the following
endpoints:

`/admin/config`

Provides a REST API to the server configuration with a JSON-based view of `cn=config` and the
configuration backend.

Each LDAP entry maps to a resource under `/admin/config`, with default values shown in the
resource even if they are not set in the LDAP representation.

`/admin/monitor`

Provides a REST API to the server monitoring information with a read-only JSON-based view of
`cn=monitor` and the monitoring backend.

Each LDAP entry maps to a resource under `/admin/monitor`.

To use the Admin endpoint APIs, follow these steps:

1. Grant users access to the endpoints as appropriate:

   - For access to `/admin/config`, assign `config-read` or `config-write` privileges.

     The following example assigns the `config-read` privilege to Kirsten Vaughan:

     ```
     $ cat config-read.ldif
     dn: uid=kvaughan,ou=People,dc=example,dc=com
     changetype: modify
     add: ds-privilege-name
     ds-privilege-name: config-read

     $ ldapmodify \
      --port 1389 \
      --bindDN "cn=Directory Manager" \
      --bindPassword password \
      config-read.ldif
     ```

     For more detail, see "Configuring Privileges".

   - For access to `/admin/monitor`, authenticated users can read information.

2. (Optional) If necessary, adjust the `authorization-mechanism` setting for the Admin endpoint.

   By default, the Admin endpoint uses the HTTP Basic authorization mechanism. The HTTP Basic authorization mechanism default configuration resolves the user identity extracted from the HTTP request to an LDAP user identity as follows:

   1. If the request has an `Authorization: Basic` header for HTTP Basic authentication, the server extracts the username and password.

   2. If the request has `X-OpenIDM-Username` and `X-OpenIDM-Password` headers, the server extracts the username and password.

   3. The server uses the default Exact Match identity mapper to search for a unique match between the username and the UID attribute value of an entry in the public naming contexts of the OpenDJ server.

      In other words, in LDAP terms, it searches under all user data base DNs for `(uid=http-username)`. The username `kvaughan` maps to the example entry with DN `uid=kvaughan,ou=People,dc=example,dc=com`.

   For details on configuring HTTP authorization mechanisms, see "To Set Up HTTP Authorization".

3. (Optional) Consider protecting traffic to the Admin endpoint by using HTTPS as described in "To Set Up HTTP Access".

4. Test access to the endpoint as an authorized user.

The examples below use the (self-signed) server certificate which the following command writes into file named `server-cert.pem`:

```
$ keytool \
 -export \
 -rfc \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -file server-cert.pem
Certificate stored in file <server-cert.pem>
```

The following example demonstrates reading the Admin endpoint resource under `/admin/config`:

```
$ curl \
 --cacert server-cert.pem \
 --user kvaughan:bribery \
 https://opendj.example.com:8443/admin/config/http-endpoints/%2Fadmin?_prettyPrint=true
{
  "_id" : "/admin",
  "_rev" : "<revision>",
  "_schema" : "admin-endpoint",
  "java-class" : "org.opends.server.protocols.http.rest2ldap.AdminEndpoint",
  "base-path" : "/admin",
  "enabled" : true,
  "authorization-mechanism" : "HTTP Basic"
}
```

Notice how the path to the resource in the example above, `/admin/config/http-endpoints/%2Fadmin`, corresponds to the DN of the entry under cn=config, which is `ds-cfg-base-path=/admin,cn=HTTP Endpoints,cn=config`.

The following example demonstrates reading everything under `/admin/monitor`:

```
$ curl \
 --cacert server-cert.pem \
 --user kvaughan:bribery \
 https://opendj.example.com:8443/admin/monitor?_queryFilter=true
```

To Set Up OpenDJ REST to LDAP Gateway

Follow these steps to set up OpenDJ REST to LDAP gateway Servlet to access your directory service.

1. Download and install the gateway as described in "To Install the REST to LDAP Gateway" in the *Installation Guide*.

2. Adjust the configuration for your directory service as described in "*REST to LDAP Configuration*" in the *Reference*.

5.9. DSML Client Access

Directory Services Markup Language (DSML) client access is implemented as a servlet that runs in a web application container.

You configure DSML client access by editing the `WEB-INF/web.xml` after you deploy the web application. In particular, you must at least set the `ldap.host` and `ldap.port` parameters if they differ from the default values, which are `localhost` and `389`.

The list of DSML configuration parameters, including those that are optional, consists of the following:

`ldap.host`

Required parameter indicating the host name of the underlying directory service. Default: `localhost`.

`ldap.port`

Required parameter indicating the LDAP port of the underlying directory service. Default: 389.

`ldap.userdn`

Optional parameter specifying the DN used by the DSML gateway to bind to the underlying directory service. Not used by default.

`ldap.userpassword`

Optional parameter specifying the password used by the DSML gateway to bind to the underlying directory service. Not used by default.

`ldap.authzidtypeisid`

This parameter can help you set up the DSML gateway to do HTTP Basic Access Authentication, given the appropriate mapping between the user ID, and the user's entry in the directory.

Required boolean parameter specifying whether the HTTP Authorization header field's Basic credentials in the request hold a plain ID, rather than a DN. If set to `true`, then the gateway performs an LDAP SASL bind using SASL plain, enabled by default in OpenDJ to look for an exact match between a `uid` value and the plain ID value from the header. In other words, if the plain ID is `bjensen`, and that corresponds in the directory service to Babs Jensen's entry with DN `uid=bjensen,ou=people,dc=example,dc=com`, then the bind happens as Babs Jensen. Note also that you can configure OpenDJ identity mappers for scenarios that use a different attribute than `uid`, such as the `mail` attribute.

Default: `false`

ldap.usessl

Required parameter indicating whether `ldap.port` points to a port listening for LDAPS (LDAP/SSL) traffic. Default: `false`.

ldap.usestarttls

Required parameter indicating whether to use StartTLS to connect to the specified `ldap.port`. Default: `false`.

ldap.trustall

Required parameter indicating whether to blindly trust all certificates presented to the DSML gateway when using secure connections (LDAPS or StartTLS). Default: `false`.

ldap.truststore.path

Optional parameter indicating the truststore used to verify certificates when using secure connections. If you want to connect using LDAPS or StartTLS, and do not want the gateway blindly to trust all certificates, then you must set up a truststore. Not used by default.
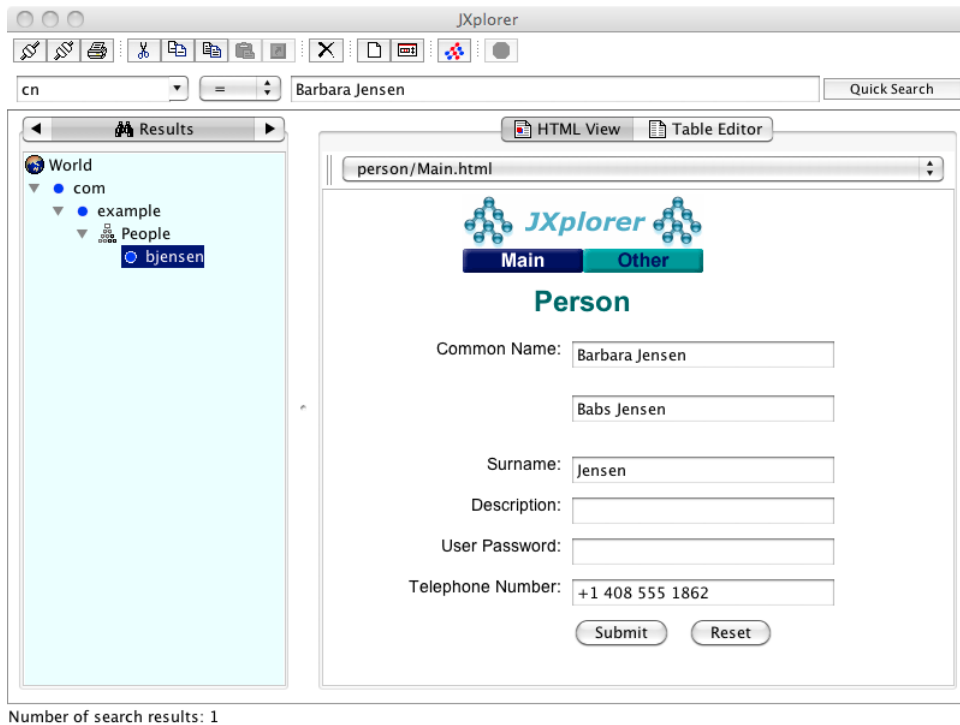
ldap.truststore.password

Optional parameter indicating the truststore password. If you set up and configure a truststore, then you need to set this as well. Not used by default.

The DSML servlet translates between DSML and LDAP, and passes requests to the directory service. For initial testing purposes, you might try JXplorer, where DSML Service: /*webapp-dir*/DSMLServlet. Here, *webapp-dir* refers to the name of the directory in which you unpacked the DSML `.war`. "JXplorer Accessing an OpenDJ Directory Service" shows the result.

JXplorer Accessing an OpenDJ Directory Service

Number of search results: 1

5.10. JMX Client Access

You configure Java Management Extensions (JMX) client access by using the command-line tool, **dsconfig**.

To Set Up JMX Access

• Configure the server to activate JMX access.

The following example uses the reserved port number, 1689:

```
$ dsconfig \
 create-connection-handler \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "JMX Connection Handler" \
 --type jmx \
 --set enabled:true \
 --set listen-port:1689 \
 --trustAll \
 --no-prompt
```

The change takes effect immediately.

To Configure Access To JMX

After you set up an OpenDJ server to listen for JMX connections, you must assign privileges in order to allow a user to connect over JMX:

1. Assign the privileges, `jmx-notify`, `jmx-read`, and `jmx-write` as necessary to the user who connects over JMX.

   For details see "Configuring Privileges".

2. Connect using the service URI, user name, and password:

   Service URI

   Full URI to the service including the hostname or IP address and port number for JMX where the OpenDJ server listens for connections.

   For example, if the server hostname is `opendj.example.com`, and the OpenDJ server listens for JMX connections on port 1689, then the service URI is `service:jmx:rmi:///jndi/rmi://opendj.example.com:1689/org.opends.server.protocols.jmx.client-unknown`.

   User name

   The full DN of the user with privileges to connect over JMX, such as `cn=My App,ou=Apps,dc=example,dc=com`.

   Password

   The bind password for the user.

5.11. LDIF File Access

The LDIF connection handler lets you make changes to directory data by placing LDIF files in a file system directory that the OpenDJ server regularly polls for changes. The LDIF, once consumed, is deleted.

To Set Up LDIF File Access

1. Add the directory where you put LDIF to be processed:

   ```
   $ mkdir /path/to/opendj/config/auto-process-ldif
   ```

   This example uses the default value of the `ldif-directory` property for the LDIF connection handler.

2. Activate LDIF file access:

   ```
   $ dsconfig \
    set-connection-handler-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --handler-name "LDIF Connection Handler" \
    --set enabled:true \
    --trustAll \
    --no-prompt
   ```

   The change takes effect immediately.

5.12. SNMP Access

For instructions on setting up the SNMP connection handler, see "SNMP-Based Monitoring".

Chapter 6

Configuring Privileges and Access Control

OpenDJ software supports these mechanisms to protect access:

- *Access control instructions* (ACI)

  Access control instructions apply to directory data, providing fine-grained control over what a user or group member is authorized to do in terms of LDAP operations. Most access control instructions specify scopes (targets) to which they apply such that an administrative user who has all access to `dc=example,dc=com` need not have any access to `dc=example,dc=org`.

- Administrative *privileges*

  Privileges control the administrative tasks that users can perform, such as bypassing the access control mechanism, performing backup and restore operations, making changes to the configuration, and other tasks.

  By default, privileges restrict administrative access to directory root users, though any user can be assigned a privilege. Privileges apply to an OpenDJ server, and do not have a scope.

- *Global access control policies*

  Global access control policies provide coarse-grained access control suitable for use on proxy servers, where the lack of local access to directory data makes ACIs a poor fit.

> **Note**
>
> Access control instructions (ACI) and global access control policies rely on different access control handlers, which implement different access control models. ACIs rely on the DSEE-compatible access control handler. (DSEE refers to Sun Java System Directory Server Enterprise Edition.) Global access control policies rely on the policy-based access control handler. A server can only use one handler at a time.
>
> Take the following constraints into consideration:
>
> - When the policy-based handler is configured, ACIs have no effect.
>
> - When the DSEE-compatible handler is configured, global access control policies have no effect.
>
> - When a server is set up as a directory server, it uses the DSEE-compatible access control handler, with ACIs in directory data and global ACIs in the configuration.
>
> - When a server is set up as a directory proxy server, it uses the policy-based access control policy handler, and global access control policies.

> • Once the server has been set up, the choice of access control handler cannot be changed with the **dsconfig** command.

This chapter covers each mechanism. In this chapter you will learn to:

• Configure privileges for directory administration

• Configure coarse-grained global access control policies

• Read and write access control instructions

• Configure access rights by setting access control instructions

• Evaluate effective access rights for a particular user

Some operations require both privileges and access control. For example, in order to reset user's passwords, an administrator needs the `password-reset` privilege and access to write `userPassword` values on user entries. By combining access control and privileges, you can effectively restrict scope of privileges to a particular branch of the Directory Information Tree.

6.1. About Privileges

Privileges provide access control for server administration independently from ACIs.

By default, directory root users, such as `cn=Directory Manager`, are granted the privileges marked with an asterisk (*). Other administrator users can be assigned these privileges, too:

`backend-backup`*

   Request a task to back up data

`backend-restore`*

   Request a task to restore data from backup

`bypass-acl`*

   Perform operations without regard to ACIs

`bypass-lockdown`*

   Perform operations without regard to lockdown mode

`cancel-request`*

   Cancel any client request

`changelog-read`*

   Read the changelog (under `cn=changelog`)

config-read*

Read the server configuration

config-write*

Change the server configuration

data-sync

Perform data synchronization

disconnect-client*

Close any client connection

jmx-notify

Subscribe to JMX notifications

jmx-read

Read JMX attribute values

jmx-write

Write JMX attribute values

ldif-export*

Export data to LDIF

ldif-import*

Import data from LDIF

modify-acl*

Change ACIs

password-reset*

Reset other users' passwords

privilege-change*

Change the privileges assigned to users

Take great care when assigning this privilege, as it allows the user to assign themselves all other administrative privileges.

proxied-auth

Use the Proxied Authorization control

server-lockdown*

Put the server into and take the server out of lockdown mode

server-restart*

Request a task to restart the server

server-shutdown*

Request a task to stop the server

subentry-write*

Perform LDAP subentry write operations

unindexed-search*

Search using a filter with no corresponding index

update-schema*

Change LDAP schema definitions

6.2. Configuring Privileges

For root directory administrators, by default `cn=Directory Manager`, you configure default privileges using the **dsconfig** command as described in "To Change Default Root DN Privileges".

For all directory administrators, you can change privileges with the **ldapmodify** command as shown in the following procedures:

- "To Add Privileges for an Individual Administrator"

- "To Add Privileges for a Group of Administrators"

- "To Limit Inherited Privileges"

To Change Default Root DN Privileges

Follow these steps to change default privileges for all Root DN administrators, whose entries are under `cn=Root DNs,cn=config`.

1. Start **dsconfig** in interactive mode:

    ```
    $ dsconfig \
     --port 4444 \
     --hostname opendj.example.com \
     --bindDN "cn=Directory Manager" \
     --bindPassword password
    ```

2. Select the Root DN menu.

3. Select View and edit the Root DN.

4. Edit the `default-root-privilege-name`.

5. Make sure you apply the changes when finished.

To Add Privileges for an Individual Administrator

Privileges are specified using the `ds-privilege-name` operational attribute, which you can change using the **ldapmodify** command.

1. Determine the privileges to add:

```
$ cat privileges.ldif
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: config-read
ds-privilege-name: password-reset
```

This example lets the user read the server configuration, and reset user passwords. In order for the user to be able to change a user password, you must also allow the modification using ACIs.

For this example, Kirsten Vaughan is a member of the Directory Administrators group for Example.com, and already has access to modify user entries.

Prior to having the privileges, Kirsten gets messages about insufficient access when trying to read the server configuration, or trying to reset a user password:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN cn=config \
 "(objectclass=*)"
# The LDAP search request failed: 50 (Insufficient Access Rights)
# Additional Information:  You do not have sufficient privileges to perform search operations in the
 Directory Server configuration
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com" \
 --newPassword changeit
The LDAP password modify operation failed: 50 (Insufficient Access Rights)
Additional Information:  You do not have sufficient privileges to perform
password reset operations
```

2. Apply the change as a user with the `privilege-change` privilege:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 privileges.ldif
```

At this point, Kirsten can perform the operations requiring privileges:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN cn=config \
 "(objectclass=*)"
dn: cn=config
…
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com" \
 --newPassword changeit
The LDAP password modify operation was successful
```

To Add Privileges for a Group of Administrators

For deployments with more than one administrator, use groups to define administrative rights.

You can use a collective attribute subentry to specify privileges for the administrator group.

Collective attributes provide a standard mechanism for defining attributes that appear on all the entries in a particular subtree. An OpenDJ server extends collective attributes to give you fine-grained control over which entries are in scope. In addition, an OpenDJ server lets you use virtual attributes, such as `isMemberOf` to construct the filter for targeting entries in scope. This allows you, for example, to define administrative privileges that apply to all users who belong to an administrator group:

1. Create an LDAP subentry that specifies the collective attributes:

```
$ cat collective.ldif
dn: cn=Administrator Privileges,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Administrator Privileges
ds-privilege-name;collective: config-read
ds-privilege-name;collective: config-write
ds-privilege-name;collective: ldif-export
ds-privilege-name;collective: modify-acl
ds-privilege-name;collective: password-reset
ds-privilege-name;collective: proxied-auth
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 collective.ldif
```

The Directory Administrators group for Example.com includes members like Harry Miller.

2. Observe that the change takes effect immediately:

```
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "uid=hmiller,ou=people,dc=example,dc=com" \
 --bindPassword hillock \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com"
The LDAP password modify operation was successful
Generated Password:  <password>
```

To Limit Inherited Privileges

When privileges are set as described in "To Add Privileges for a Group of Administrators", the same list of privileges is applied to every target account. An OpenDJ server also assigns default directory root DN privileges as described in "To Change Default Root DN Privileges".

In some cases, the list of inherited privileges can be too broad. OpenDJ servers have a mechanism to limit effective privileges by preceding the privilege attribute value with a `-`.

The following steps show how to prevent Kirsten Vaughan from resetting passwords when the privilege is assigned as described in "To Add Privileges for a Group of Administrators":

1. Check the privilege settings for the account:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 ds-privilege-name
dn: uid=kvaughan,ou=People,dc=example,dc=com
ds-privilege-name: config-read
ds-privilege-name: password-reset
```

2. Set the privilege attribute for the account to deny the privilege:

```
$ cat restrict-privileges.ldif
dn: uid=kvaughan,ou=people,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: -password-reset

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 restrict-privileges.ldif
```

3. Observe that the privilege is no longer in effect:

```
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com"
The LDAP password modify operation failed: 50 (Insufficient Access Rights)
Additional Information:  You do not have sufficient privileges to perform
password reset operations
```

6.3. About ACIs

OpenDJ server ACIs exist as operational `aci` attribute values on directory entries, and as global ACI attributes stored in the configuration. ACIs apply to a scope defined in the instruction. They set permissions that depend on what operation is requested, who requested the operation, and how the client connected to the server.

For example, the ACIs on the following entry allow anonymous read access to all attributes except passwords, and allow read-write access for directory administrators under `dc=example,dc=com`:

```
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example
aci: (target ="ldap:///dc=example,dc=com")
 (targetattr != "userPassword")(version 3.0;acl "Anonymous read-search access";
 allow(read, search, compare)(userdn = "ldap:///anyone");)
aci: (target="ldap:///dc=example,dc=com")
 (targetattr = "*")(version 3.0; acl "allow all Admin group";
 allow(all) groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
```

An OpenDJ server's default behavior is that no access is allowed unless it is specifically granted by an ACI. (The `bypass-acl` privilege, assigned to certain users such as `cn=Directory Manager`, can allow users to bypass access control checks.)

An OpenDJ server provides default global ACIs to facilitate evaluation while maintaining a reasonable security policy. By default, users are granted access to:

• Read the root DSE

• Read the LDAP schema

• Use certain LDAP controls and extended operations

• Modify their own entries

• Read public data and operational attributes

Global ACIs are defined on the access control handler, and apply to the entire server. You must adjust the default global ACIs to match the security policies for your organization, for example, to restrict anonymous access.

ACI attribute values use a specific language described in this section. Although ACI attribute values can become difficult to read in LDIF, the basic syntax is simple:

```
targets(version 3.0;acl "name";permissions subjects;)
```

The following list briefly explains the variables in the syntax above:

targets

The *targets* specifies entries, attributes, controls, and extended operations to which the ACI applies.

To include multiple *targets*, enclose each individual target in parentheses, `()`. When you specify multiple targets, all targets must match for the ACI to apply (`AND`).

name

Supplies a human-readable description of what the ACI does.

permissions

Defines which actions to allow, and which to deny. Paired with *subjects*.

subjects

Identifies clients to which the ACI applies depending on who connected, and when, where, and how they connected. Paired with *permissions*.

Separate multiple pairs of *permissions-subjects* definitions with semicolons, `;`. When you specify multiple permissions-subjects pairs, at least one must match (`OR`).

6.3.1. ACI Targets

The seven types of ACI targets identify the objects to which the ACI applies. Most expressions allow you to use either `=` to specify that the target should match the value or `!=` to specify that the target should not match the value:

`(target [!]= "ldap:///DN")`

Sets the scope to the entry with distinguished name *DN*, and to child entries.

You can use asterisks, `*`, to replace attribute types, attribute values, and entire DN components.

In other words, the following specification targets both `uid=bjensen,ou=People,dc=example,dc=com` and `cn=My App,ou=Apps,dc=example,dc=com`:

```
(target = "ldap:///*=*,*,dc=example,dc=com")
```

The *DN* must be in the subtree of the entry where the ACI is defined.

If you do not specify `target`, then the entry holding this ACI is affected. If `targetscope` is also omitted, then this entry and all subordinates are affected.

`(targetattr [!]= "attr-list")`

Replace *attr-list* with a list of attribute type names, such as `userPassword`, separating multiple attribute type names with `||`.

This specification affects the entry where the ACI is located, or the entries specified by other targets in the ACI.

You can use an asterisk, `*`, to specify all user attributes, although you will see better performance when explicitly including or excluding attribute types as needed. You can use a plus sign, `+`, to specify all operational attributes.

A negated *attr-list* of operational attributes will only match other operational attributes and never any user attributes, and vice-versa.

If you do not include this target specification, then by default no attributes are affected by the ACI.

(targetfilter [!]= "*ldap-filter*")

Sets the scope to match the *ldap-filter* dynamically, as in an LDAP search. The *ldap-filter* can be any valid LDAP filter.

(targattrfilters [!]= "*expression*")

Use this target specification when managing changes made to particular attributes.

The *expression* takes one of the following forms. Separate expressions with semicolons (`;`):

```
op=attr1:filter1[&& attr2:filter2 …][;op=attr3:filter3[&& attr4:filter4 …] …]
```

The *op* can be either `add` for operations creating attributes, or `del` for operations removing them.

Replace *attr* with an attribute type. Replace *filter* with an LDAP filter that corresponds to the *attr* attribute type.

(targetscope = "base|onelevel|subtree|subordinate")

- `base` refers to the entry with the ACI.

- `onelevel` refers to immediate children.

- `subtree` refers to the base entry and all children.

- `subordinate` refers to all children only.

If you do not specify `targetscope`, the default is `subtree`.

(targetcontrol [!]= "*OID*")

Replace *OID* with the object identifier for the LDAP control to target. Separate multiple OIDs with `||`.

To use an LDAP control, the bind DN user must have `allow(read)` permissions.

This target cannot be restricted to a specific subtree by combining it with another target.

(extop [!]= "*OID*")

Replace *OID* with the object identifier for the extended operation to target. Separate multiple OIDs with `||`.

To use an LDAP extended operation, the bind DN user must have `allow(read)` permissions.

This target cannot be restricted to a specific subtree by combining it with another target.

6.3.2. ACI Permissions

ACI permission definitions take one of the following forms:

```
allow(action[, action …])
```

```
deny(action[, action …])
```

> **Tip**
>
> Although deny is supported, avoid restricting permissions by using deny. Instead, explicitly allow access only as needed. What looks harmless and simple in tests and examples can grow difficult to maintain in a real-world deployment with nested ACIs.

Replace *action* with one of the following:

add

Entry creation, as for an LDAP add operation.

all

All permissions, except export, import, proxy.

compare

Attribute value comparison, as for an LDAP compare operation.

delete

Entry deletion, as for an LDAP delete operation.

export

Entry export during a modify DN operation.

Despite the name, this action is unrelated to LDIF export operations.

import

Entry import during a modify DN operation.

Despite the name, this action is unrelated to LDIF import operations.

proxy

Access the ACI target using the rights of another user.

read

Read entries and attributes, or use an LDAP control or extended operation.

search

> Search the ACI targets.
>
> Combine with `read` to read the search results.

selfwrite

> Add or delete own DN from a group.

write

> Modify attributes on ACI target entries.

6.3.3. ACI Subjects

ACI subjects match characteristics of the client connection to the server. Use subjects to restrict whether the ACI applies depending on who connected, and when, where, and how they connected. Most expressions allow you to use either `=` to specify that the condition should match the value or `!=` to specify that the condition should not match the value:

authmethod [!]= "none|simple|ssl|sasl *mech*"

> - `none` means do not check.
>
> - `simple` means simple authentication.
>
> - `ssl` refers to certificate-based authentication over LDAPS.
>
> - `sasl *mech*` refers to SASL, where *mech* is DIGEST-MD5, EXTERNAL, or GSSAPI.

dayofweek [!]= "*day*[, *day* …]"

> Replace *day* with one of:
>
> ```
> sun
> mon
> tue
> wed
> thu
> fri
> sat
> ```

dns [!]= "*hostname*"

> Use asterisks, `*`, to replace name components, such as `dns = "*.example.com"`.

groupdn [!]= "ldap:///*DN*[|| ldap:///*DN* …]"

> Replace *DN* with the distinguished name of a group to permit or restrict access for members.

ip [!]= "*addresses*"

The *addresses* can be specified for IPv4 or IPv6.

IPv6 addresses are specified in brackets as `ldap://[address]/subnet-prefix` where /*subnet-prefix* is optional.

You can specify:

- Individual IPv4 addresses

- Addresses with asterisks (`*`) to replace subnets and host numbers

- CIDR notation

- Forms such as `192.168.0.*+255.255.255.0` to specify subnet masks

ssf = "*strength*"
ssf != "*strength*"
ssf > "*strength*"
ssf >= "*strength*"
ssf < "*strength*"
ssf <= "*strength*"

The security strength factor (`ssf`) pertains to the cipher key strength for connections using DIGEST-MD5, GSSAPI, SSL, or TLS.

For example, to require that the connection must have a cipher strength of at least 256 bits, specify `ssf >= "256"`.

timeofday = "*hhmm*"
timeofday != "*hhmm*"
timeofday > "*hhmm*"
timeofday >= "*hhmm*"
timeofday < "*hhmm*"
timeofday <= "*hhmm*"

The *hhmm* is expressed as on a 24-hour clock.

For example, 1:15 PM is written `1315`.

userattr [!]= "*attr#value*"
userattr [!]= *ldap-url*#LDAPURL"
userattr [!]= "[parent[*child-level*].]*attr*#GROUPDN|USERDN"

The `userattr` subject specifies an attribute that must match on both the bind entry and the target of the ACI.

To match when the user attribute on the bind DN entry corresponds directly to the attribute on the target entry, replace *attr* with the user attribute type, and *value* with the attribute value. An

OpenDJ server performs an internal search to get the attributes of the bind entry. Therefore, this ACI subject does not work with operational attributes.

To match when the target entry is identified by an LDAP URL, and the bind DN is in the subtree of the DN of the LDAP URL, use `ldap-url#LDAPURL`.

To match when the bind DN corresponds to a member of the group identified by the *attr* value on the target entry, use `attr#GROUPDN`.

To match when the bind DN corresponds to the *attr* value on the target entry, use `attr#USERDN`.

The optional inheritance specification, `parent[child-level].`, lets you specify how many levels below the target entry inherit the ACI. The *child-level* is a number from 0 to 9, with 0 indicating the target entry only. Separate multiple *child-level* digits with commas (`,`).

`userdn [!]= "ldap-url++[|| ldap-url++ …]"`

To match the bind DN, replace *ldap-url++* with either a valid LDAP URL, such as `ldap:///uid=bjensen,ou=People,dc=example,dc=com`, or `ldap:///dc=example,dc=com??sub?(uid=bjensen)`, or a special LDAP URL-like keyword from the following list:

`ldap:///all`

> Match authenticated users.

`ldap:///anyone`

> Match anonymous and authenticated users.

`ldap:///parent`

> Match when the bind DN is a parent of the ACI target.

`ldap:///self`

> Match when the bind DN entry corresponds to ACI target.

6.3.4. How ACI is Evaluated

Understanding how an OpenDJ server evaluates the `aci` values is critical when implementing an access control policy.

The rules the server follows are simple:

1. To determine whether an operation is allowed or denied, an OpenDJ server looks in the directory for the target of the operation. It collects any ACI values from that entry, and then walks up the directory tree to the suffix, collecting all ACI values en route. Global ACI values are then collected.

2. It then separates the ACI values into two lists; one list contains all the ACI values that match the target and deny the required access, and the other list contains all the ACI values that match the target and allow the required access.

3. If the deny list contains any ACI values after this procedure, access is immediately denied.

4. If the deny list is empty, then the allow list is processed. If the allow list contains any ACI values, access is allowed.

5. If both lists are empty, access is denied.

Note

Some operations require multiple permissions and involve multiple targets. Evaluation will therefore take place multiple times. For example, a search operation requires the `search` permission for each attribute in the search filter. If all those are allowed, the `read` permission is used to decide what attributes and values can be returned.

6.3.5. ACI Required For LDAP Operations

The minimal access control information required for specific LDAP operations is described here:

Add

The ACI must allow the `add` permission to entries in the target. This implicitly allows the attributes and values to be set.

Use `targattrfilters` to explicitly deny access to any values if required.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to add an entry is:

```
aci: (version 3.0;acl "Add entry"; allow (add)
  (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Bind

Because this is used to establish the user's identity and derived authorizations, ACI is irrelevant for this operation and is not checked.

To prevent authentication, disable the account instead. For details, see "Managing Accounts Manually".

Compare

The ACI must allow the `compare` permission to the attribute in the target entry.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to compare values against the `sn` attribute is:

```
aci: (targetattr = "sn")(version 3.0;acl "Compare surname"; allow (compare)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Delete

The ACI must allow the `delete` permission to the target entry. This implicitly allows the attributes and values in the target to be deleted.

Use `targattrfilters` to explicitly deny access to the values if required.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to delete an entry is:

```
aci: (version 3.0;acl "Delete entry"; allow (delete)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Modify

The ACI must allow the `write` permission to attributes in the target entries. This implicitly allows all values in the target attribute to be modified.

Use `targattrfilters` to explicitly deny access to specific values if required.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to modify the `description` attribute in an entry is:

```
aci: (targetattr = "description")(version 3.0; acl "Modify description"; allow (write)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

ModifyDN

If the entry is being moved to a `newSuperior`, the `export` permission must be allowed on the target, and the `import` permission must be allowed on the `newSuperior` entry.

The ACI must allow `write` permission to the attributes in the old RDN and the new RDN. All values of the old RDN and new RDN can be written implicitly; use `targattrfilters` to explicitly deny access to values used if required.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to rename entries named with the `uid` attribute to new locations:

```
aci: (targetattr = "uid")(version 3.0;acl "Rename uid= entries";
 allow (write, import, export)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Search

ACI is required to process the search filter, and to determine what attributes and values may be returned in the results. The `search` permission is used to allow particular attributes in the search filter. The `read` permission is used to allow particular attributes to be returned.

If `read` permission is allowed to any attribute, the server automatically allows the `objectClass` attribute to also be read.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to search for `uid` attributes, and also to read that attribute in matching entries is:

```
aci: (targetattr = "uid")(version 3.0;acl "Search and read uid"; allow (search, read)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Use Control or Extended Operation

The ACI must allow the `read` permission to the `targetcontrol` or `extop` OIDs.

For example, the ACI required to allow `uid=bjensen,ou=People,dc=example,dc=com` to use the Persistent Search request control with OID `2.16.840.1.113730.3.4.3` is:

```
aci: (targetcontrol = "2.16.840.1.113730.3.4.3")
 (version 3.0;acl "Request Persistent Search"; allow (read)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

6.4. Configuring ACIs

ACIs are defined in the data on `aci` attributes. They can be imported in LDIF and modified over LDAP. In order to make changes to ACIs, however, users first need the `modify-acl` privilege. By default, only the root DN user has the `modify-acl` privilege.

Global ACIs on `cn=Access Control Handler,cn=config` can be set using the **dsconfig** command. Global ACIs have attribute type `ds-cfg-global-aci`.

Modifying and removing global ACIs can have deleterious effects. Generally the impact depends on your deployment requirements. Modifications to global ACIs fall into the following categories:

• Modification or removal is permitted.

   You must test client applications when deleting the specified ACI.

• Modification or removal may affect applications.

   You must test client applications when modifying or deleting the specified ACI.

• Modification or removal may affect applications, but is not recommended.

You must test client applications when modifying or deleting the specified ACI.

• Do not modify or delete.

For details, see

Default Global ACIs

Name	Description	ACI Definition
Anonymous control access	Anonymous and authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	(targetcontrol="2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 || 2.16.840 .1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10 .2 || 1.3.6.1.4.1.42.2.27.8.5.1 || 2.16 .840.1.113730.3.4.16 || 1.2.840.113556.1 .4.1413 || 1.3.6.1.4.1.36733.2.1.5.1") (version 3.0; acl "Anonymous control access"; allow(read) userdn="ldap:/// anyone";)
Anonymous control access	Anonymous and authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	(targetcontrol="2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 || 2.16.840 .1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10 .2 || 1.3.6.1.4.1.42.2.27.8.5.1 || 2.16 .840.1.113730.3.4.16 || 1.2.840.113556.1 .4.1413 || 1.3.6.1.4.1.36733.2.1.5.1") (version 3.0; acl "Anonymous control access"; allow(read) userdn="ldap:/// anyone";)
Anonymous control access	Anonymous and authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	(targetcontrol="2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 || 2.16.840 .1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10 .2 || 1.3.6.1.4.1.42.2.27.8.5.1 || 2.16 .840.1.113730.3.4.16 || 1.2.840.113556.1 .4.1413 || 1.3.6.1.4.1.36733.2.1.5.1") (version 3.0; acl "Anonymous control access"; allow(read) userdn="ldap:/// anyone";)
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID. Modification or removal may affect applications.	(extop="1.3.6.1.4.1.26027.1.6.1 || 1.3.6 .1.4.1.26027.1.6.3 || 1.3.6.1.4.1.4203.1 .11.1 || 1.3.6.1.4.1.1466.20037 || 1.3.6 .1.4.1.4203.1.11.3") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:///anyone";)
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID. Modification or removal may affect applications.	(extop="1.3.6.1.4.1.26027.1.6.1 || 1.3.6 .1.4.1.26027.1.6.3 || 1.3.6.1.4.1.4203.1 .11.1 || 1.3.6.1.4.1.1466.20037 || 1.3.6 .1.4.1.4203.1.11.3") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:///anyone";)

Name	Description	ACI Definition
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID. Modification or removal may affect applications.	`(extop="1.3.6.1.4.1.26027.1.6.1 || 1.3.6.1.4.1.26027.1.6.3 || 1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037 || 1.3.6.1.4.1.4203.1.11.3") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:///anyone";)`
Anonymous read access	Anonymous and authenticated users can read the user data attributes that are specified by their names. Modification or removal is permitted.	`(targetattr!="userPassword||authPassword||debugsearchindex||changes||changeNumber||changeType||changeTime||targetDN||newRDN||newSuperior||deleteOldRDN")(version 3.0; acl "Anonymous read access"; allow (read,search,compare) userdn="ldap:///anyone";)`
Anonymous read access	Anonymous and authenticated users can read the user data attributes that are specified by their names. Modification or removal is permitted.	`(targetattr!="userPassword||authPassword||debugsearchindex||changes||changeNumber||changeType||changeTime||targetDN||newRDN||newSuperior||deleteOldRDN")(version 3.0; acl "Anonymous read access"; allow (read,search,compare) userdn="ldap:///anyone";)`
Anonymous read access	Anonymous and authenticated users can read the user data attributes that are specified by their names. Modification or removal is permitted.	`(targetattr!="userPassword||authPassword||debugsearchindex||changes||changeNumber||changeType||changeTime||targetDN||newRDN||newSuperior||deleteOldRDN")(version 3.0; acl "Anonymous read access"; allow (read,search,compare) userdn="ldap:///anyone";)`
Authenticated users control access	Authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	`(targetcontrol="1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 || 1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 || 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 || 1.3.6.1.4.1.42.2.27.9.5.9") (version 3.0; acl "Authenticated users control access"; allow(read) userdn="ldap:///all";)`
Authenticated users control access	Authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	`(targetcontrol="1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 || 1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 || 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 || 1.3.6.1.4.1.42.2.27.9.5.9") (version 3.0; acl "Authenticated users control`

Name	Description	ACI Definition
		access"; allow(read) userdn="ldap:/// all";)
Authenticated users control access	Authenticated users can use the LDAP controls that are specified by OID. Modification or removal may affect applications.	(targetcontrol="1.3.6.1.1.12 || 1.3.6.1.1 .13.1 || 1.3.6.1.1.13.2 || 1.2.840.113556 .1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 || 2.16.840.1 .113730.3.4.9 || 1.2.840.113556.1.4.473 || 1.3.6.1.4.1.42.2.27.9.5.9") (version 3.0; acl "Authenticated users control access"; allow(read) userdn="ldap:/// all";)
Self entry modification	Authenticated users can modify the specified attributes on their own entries. Modification or removal is permitted.	(targetattr="audio||authPassword|| description||displayName||givenName|| homePhone||homePostalAddress||initials|| jpegPhoto||labeledURI||mobile|| pager||postalAddress||postalCode|| preferredLanguage||telephoneNumber|| userPassword")(version 3.0; acl "Self entry modification"; allow (write) userdn="ldap:///self";)
Self entry modification	Authenticated users can modify the specified attributes on their own entries. Modification or removal is permitted.	(targetattr="audio||authPassword|| description||displayName||givenName|| homePhone||homePostalAddress||initials|| jpegPhoto||labeledURI||mobile|| pager||postalAddress||postalCode|| preferredLanguage||telephoneNumber|| userPassword")(version 3.0; acl "Self entry modification"; allow (write) userdn="ldap:///self";)
Self entry modification	Authenticated users can modify the specified attributes on their own entries. Modification or removal is permitted.	(targetattr="audio||authPassword|| description||displayName||givenName|| homePhone||homePostalAddress||initials|| jpegPhoto||labeledURI||mobile|| pager||postalAddress||postalCode|| preferredLanguage||telephoneNumber|| userPassword")(version 3.0; acl "Self entry modification"; allow (write) userdn="ldap:///self";)
Self entry read	Authenticated users can read the password values on their own entries. By default, the server applies a one-way hash algorithm to the password value before writing it to the entry, so it is computationally difficult to recover the cleartext version of the password from the stored value. Modification or removal is permitted.	(targetattr="userPassword||authPassword") (version 3.0; acl "Self entry read"; allow (read,search,compare) userdn="ldap:///self";)

Name	Description	ACI Definition
Self entry read	Authenticated users can read the password values on their own entries. By default, the server applies a one-way hash algorithm to the password value before writing it to the entry, so it is computationally difficult to recover the cleartext version of the password from the stored value. Modification or removal is permitted.	`(targetattr="userPassword||authPassword")` `(version 3.0; acl "Self entry` `read"; allow (read,search,compare)` `userdn="ldap:///self";)`
Self entry read	Authenticated users can read the password values on their own entries. By default, the server applies a one-way hash algorithm to the password value before writing it to the entry, so it is computationally difficult to recover the cleartext version of the password from the stored value. Modification or removal is permitted.	`(targetattr="userPassword||authPassword")` `(version 3.0; acl "Self entry` `read"; allow (read,search,compare)` `userdn="ldap:///self";)`
User-Visible Operational Attributes	Anonymous and authenticated users can read attributes that identify entries and that contain information about modifications to entries. Modification or removal may affect applications.	`(targetattr="createTimestamp||` `creatorsName||modifiersName||` `modifyTimestamp||entryDN||` `entryUUID||subschemaSubentry||` `etag||governingStructureRule||` `structuralObjectClass||hasSubordinates||` `numSubordinates||isMemberOf")(version` `3.0; acl "User-Visible Operational` `Attributes"; allow (read,search,compare)` `userdn="ldap:///anyone";)`
User-Visible Operational Attributes	Anonymous and authenticated users can read attributes that identify entries and that contain information about modifications to entries. Modification or removal may affect applications.	`(targetattr="createTimestamp||` `creatorsName||modifiersName||` `modifyTimestamp||entryDN||` `entryUUID||subschemaSubentry||` `etag||governingStructureRule||` `structuralObjectClass||hasSubordinates||` `numSubordinates||isMemberOf")(version` `3.0; acl "User-Visible Operational` `Attributes"; allow (read,search,compare)` `userdn="ldap:///anyone";)`
User-Visible Operational Attributes	Anonymous and authenticated users can read attributes that identify entries and that contain information about modifications to entries. Modification or removal may affect applications.	`(targetattr="createTimestamp||` `creatorsName||modifiersName||` `modifyTimestamp||entryDN||` `entryUUID||subschemaSubentry||` `etag||governingStructureRule||` `structuralObjectClass||hasSubordinates||` `numSubordinates||isMemberOf")(version` `3.0; acl "User-Visible Operational` `Attributes"; allow (read,search,compare)` `userdn="ldap:///anyone";)`

Name	Description	ACI Definition
User-Visible Root DSE Operational Attributes	Anonymous and authenticated users can read attributes that describe what the server supports. Modification or removal may affect applications.	`(target="ldap:///")(targetscope="base") (targetattr="objectClass|| namingContexts|| supportedAuthPasswordSchemes|| supportedControl||supportedExtension|| supportedFeatures||supportedLDAPVersion|| supportedSASLMechanisms|| supportedTLSCiphers|| supportedTLSProtocols||vendorName|| vendorVersion")(version 3.0; acl "User-Visible Root DSE Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`
User-Visible Root DSE Operational Attributes	Anonymous and authenticated users can read attributes that describe what the server supports. Modification or removal may affect applications.	`(target="ldap:///")(targetscope="base") (targetattr="objectClass|| namingContexts|| supportedAuthPasswordSchemes|| supportedControl||supportedExtension|| supportedFeatures||supportedLDAPVersion|| supportedSASLMechanisms|| supportedTLSCiphers|| supportedTLSProtocols||vendorName|| vendorVersion")(version 3.0; acl "User-Visible Root DSE Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`
User-Visible Root DSE Operational Attributes	Anonymous and authenticated users can read attributes that describe what the server supports. Modification or removal may affect applications.	`(target="ldap:///")(targetscope="base") (targetattr="objectClass|| namingContexts|| supportedAuthPasswordSchemes|| supportedControl||supportedExtension|| supportedFeatures||supportedLDAPVersion|| supportedSASLMechanisms|| supportedTLSCiphers|| supportedTLSProtocols||vendorName|| vendorVersion")(version 3.0; acl "User-Visible Root DSE Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`
User-Visible Schema Operational Attributes	Anonymous and authenticated users can read LDAP schema definitions. Modification or removal may affect applications.	`(target="ldap:///cn=schema") (targetscope="base") (targetattr="objectClass|| attributeTypes||dITContentRules|| dITStructureRules||ldapSyntaxes|| matchingRules||matchingRuleUse|| nameForms||objectClasses")(version 3.0; acl "User-Visible Schema Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`

Name	Description	ACI Definition
User-Visible Schema Operational Attributes	Anonymous and authenticated users can read LDAP schema definitions. Modification or removal may affect applications.	`(target="ldap:///cn=schema")` `(targetscope="base")` `(targetattr="objectClass||` `attributeTypes||dITContentRules||` `dITStructureRules||ldapSyntaxes||` `matchingRules||matchingRuleUse||` `nameForms||objectClasses")(version 3.0;` ` acl "User-Visible Schema Operational` ` Attributes"; allow (read,search,compare)` ` userdn="ldap:///anyone";)`
User-Visible Schema Operational Attributes	Anonymous and authenticated users can read LDAP schema definitions. Modification or removal may affect applications.	`(target="ldap:///cn=schema")` `(targetscope="base")` `(targetattr="objectClass||` `attributeTypes||dITContentRules||` `dITStructureRules||ldapSyntaxes||` `matchingRules||matchingRuleUse||` `nameForms||objectClasses")(version 3.0;` ` acl "User-Visible Schema Operational` ` Attributes"; allow (read,search,compare)` ` userdn="ldap:///anyone";)`

Users with write access to add ACIs and with the `modify-acl` privilege can use the **ldapmodify** command to change ACIs located in user data.

This section focuses on ACI examples, rather than demonstrating how to update the directory for each example. To update ACIs, either change them using the **ldapmodify** command, or using OpenDJ control panel.

If you use OpenDJ control panel, find the entry to modify in the Manage Entries window. Then try View > LDIF View to edit the entry. The control panel checks ACI syntax and identifies errors before saving any changes.

For hints on updating directory entries with the **ldapmodify** command, see "Modifying Entry Attributes" in the *Developer's Guide*. Keep in mind that the name of the ACI attribute is `aci` as shown in the examples that follow.

This section includes the following examples:

- "ACI: Anonymous Reads and Searches"

- "ACI: Disable Anonymous Access"

- "ACI: Full Access for Administrators"

- "ACI: Change Your Password"

- "ACI: Manage Your Group Membership"

- "ACI: Manage Self-Service Groups"

- "ACI: Permit Cleartext Access Over Loopback Only"

- "ACI: Manage ACI Values"

ACI: Anonymous Reads and Searches

This ACI makes all user attributes world-readable except password attributes:

```
aci: (target ="ldap:///dc=example,dc=com")
  (targetattr != "authPassword || userPassword")
  (version 3.0;acl "Anonymous read-search access";
  allow (read, search, compare)(userdn = "ldap:///anyone");)
```

ACI: Disable Anonymous Access

By default, an OpenDJ server denies access unless an ACI explicitly allows access. (This does not apply to the root DN user, `cn=Directory Manager`, who has the `bypass-acl` privilege.) By default, an OpenDJ server allows anonymous users to read public data and request certain controls and extended operations.

These default capabilities are defined in global ACIs, which you can read by using the following command:

```
$ dsconfig \
 get-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --property global-aci \
 --trustAll
```

You can disable anonymous access either by editing relevant `global-aci` properties, or by using the global server configuration property, `reject-unauthenticated-requests`. Editing relevant `global-aci` properties lets you take a fine-grained approach to limit anonymous access. Setting `reject-unauthenticated-requests:true` causes an OpenDJ server to reject all requests from clients who are not authenticated except bind requests and StartTLS requests.

To take a fine-grained approach, use the **dsconfig** command to edit `global-aci` properties. One of the most expedient ways to do this is to use the command interactively on one OpenDJ server, capturing the output to a script with the `--commandFilePath` *script* option, and then editing the script for use on other servers. With this approach, you can allow anonymous read access to the root DSE and to directory schemas so that clients do not have to authenticate to discover server capabilities, and also allow anonymous users access to request certain controls and extended operations:

```
#
# Edit Access Control Handler global-aci attributes, replacing
# userdn="ldap:///anyone" (anonymous) with userdn="ldap:///all" (authenticated)
# in "Anonymous read access" and "User-Visible Operational Attributes" ACIs.
#
# To make this change, you remove existing values, and add edited values:
#
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --remove global-aci:\(targetattr!=\"userPassword\|\|authPassword\|\|debugsearchindex\|\|changes\|\|
changeNumber\|\|changeType\|\|changeTime\|\|targetDN\|\|newRDN\|\|newSuperior\|\|deleteOldRDN\"\)\(version
\ 3.0\;\ acl\ \"Anonymous\ read\ access\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///anyone\"\;\)
 \
 --remove global-aci:\(targetattr=\"createTimestamp\|\|creatorsName\|\|modifiersName\|\|modifyTimestamp\|
\|entryDN\|\|entryUUID\|\|subschemaSubentry\|\|etag\|\|governingStructureRule\|\|structuralObjectClass\|
\|hasSubordinates\|\|numSubordinates\|\|isMemberOf\"\)\(version\ 3.0\;\ acl\ \"User-Visible\ Operational\
 Attributes\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///anyone\"\;\) \
 --add global-aci:\(targetattr!=\"userPassword\|\|authPassword\|\|debugsearchindex\|\|changes\|\|
changeNumber\|\|changeType\|\|changeTime\|\|targetDN\|\|newRDN\|\|newSuperior\|\|deleteOldRDN\"\)\(version
\ 3.0\;\ acl\ \"Anonymous\ read\ access\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///all\"\;\) \
 --add global-aci:\(targetattr=\"createTimestamp\|\|creatorsName\|\|modifiersName\|\|modifyTimestamp\|\|
entryDN\|\|entryUUID\|\|subschemaSubentry\|\|etag\|\|governingStructureRule\|\|structuralObjectClass\|\|
hasSubordinates\|\|numSubordinates\|\|isMemberOf\"\)\(version\ 3.0\;\ acl\ \"User-Visible\ Operational\
 Attributes\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///all\"\;\) \
 --trustAll \
 --no-prompt
```

Make sure that you also set appropriate ACIs on any data that you import. The following commands
prevent anonymous reads of Example.com data:

```
$ cat remove-anon.ldif
dn: dc=example,dc=com
changetype: modify
delete: aci
aci: (target ="ldap:///dc=example,dc=com")(targetattr !="userPassword")(version 3
.0;acl "Anonymous read-search access";allow (read, search, compare)(userdn = "ldap:///
anyone");)
-
add: aci
aci: (target ="ldap:///dc=example,dc=com")(targetattr !="userPassword")(version 3.0;acl "Anonymous read-
search access";allow (read, search, compare)(userdn = "ldap:///all");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 remove-anon.ldif
```

At this point, clients must authenticate to view search results:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)"
$ ldapsearch  \
 --port 1389 \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" cn uid
dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
uid: bjensen
```

To reject anonymous access except bind and StartTLS requests, set `reject-unauthenticated-requests:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set reject-unauthenticated-requests:true \
 --trustAll \
 --no-prompt
```

Once you set the property, anonymous clients trying to search, for example, get an `Unwilling to Perform` response from an OpenDJ server:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)"
# The LDAP search request failed: 53 (Unwilling to Perform)
# Additional Information:  Rejecting the requested operation because the connection has not been
 authenticated
```

In both cases, notice that the changes apply to a single OpenDJ server configuration, and so are never replicated to other servers. You must apply the changes separately to each replicated server.

ACI: Full Access for Administrators

The following ACI gives a group of Directory Administrators full access. Some administrative operations will also require privileges not shown in this example:

```
aci: (target="ldap:///dc=example,dc=com")
 (targetattr = "* || +")
 (version 3.0;acl "Admins can run amok";
  allow(all, proxy, import, export)
  groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
```

`targetattr = "* || +"` permits access to all user attributes and all operational attributes. `allow(all, proxy, import, export)` permits all user operations, proxy authorization, and modify DN operations.

ACI: Change Your Password

By default, the right to change one's own password is set in a global ACI. This ACI reproduces the same effect:

```
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr = "authPassword || userPassword")
  (version 3.0;acl "Allow users to change pass words";
    allow (write)(userdn = "ldap:///self");)
```

ACI: Manage Your Group Membership

For some static groups, such as carpoolers and social club members, you might choose to let users manage their own memberships. The following ACI lets members of self-service groups manage their own membership:

```
aci: (target ="ldap:///ou=Self Service,ou=Groups,dc=example,dc=com")
  (targetattr = "member")
  (version 3.0;acl "Self registration"; allow(selfwrite)
  (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
```

ACI: Manage Self-Service Groups

This ACI lets users create and delete self-managed groups:

```
aci: (target ="ldap:///ou=Self Service,ou=Groups,dc=example,dc=com")
  (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
  (version 3.0; acl "All can create self service groups";
    allow (add)(userdn= "ldap:///uid=*,ou=People,dc=example,dc=com");)
aci: (target ="ldap:///ou=Self Service,ou=Groups,dc=example,dc=com")
  (version 3.0; acl "Owner can delete self service groups";
    allow (delete)(userattr= "owner#USERDN");)
```

ACI: Permit Cleartext Access Over Loopback Only

This ACI uses IP address and Security Strength Factor subjects:

```
aci: (target = "ldap:///dc=example,dc=com")
  (targetattr = "*")
  (version 3.0;acl "Use loopback only for LDAP in the clear";
    deny (all)(ip != "127.0.0.1" and ssf <= "1");)
```

When you use TLS but have not configured a cipher, `ssf` is one. Packets are checksummed for integrity checking, but all content is sent in cleartext.

ACI: Manage ACI Values

In order to update ACIs in directory data, a user must have both the `modify-acl` privilege and access to modify the `aci` operational attribute.

This ACI lets Directory Administrators manage `aci` values:

```
aci: (target = "ldap:///dc=example,dc=com")
 (targetattr = "aci") (version 3.0;acl "Modify ACIs"; allow (all)
 (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
```

6.5. About Global Access Control Policies

Global access control policies are similar to ACIs, but are implemented as entries in the server configuration, rather than attributes on entries. They are managed using the **dsconfig** command.

Unlike ACIs, policies can only allow access, not deny it. This constraint makes them easier to read and to change in isolation. Policies are, however, applied in addition to ACIs and privileges. It is still possible that an ACI could deny something allowed by a policy.

By default, no access is allowed until permitted by a policy or other access control. By default, a policy matches all entries, all types of connection, and all users. You set the properties of the policy to restrict its scope of application. Policies can have the settings to allow the following:

• Requests for specified LDAP controls and extended operations

• Access to specific attributes, with support for wildcards, @objectclass notation, and exceptions to simplify settings

• Read access (for read, search, and compare operations)

• Write access (for add, delete, modify, and modify DN operations)

• Making authentication required or not before requesting an operation

• Requests targeting a particular scope, with wildcards to simplify settings

• Requests originating or not from specific client addresses or domains

• Requests using a specified protocol

• Requests using a specified port

• Requests using a minimum security strength factor

• Requests from a user whose DN does or does not match a DN pattern

For details, see the reference documentation concerning Global access control policies.

6.6. Configuring Global Access Control Policies

You manage global access control policies with the **dsconfig** command. As the policies are part of the server configuration, they are not replicated, and must be configured on each server.

This section includes the following examples:

- "Example Policy: Rejecting Unauthenticated Requests"

- "Example Policy: Require Secure Connections"

- "Example Policy: Allow Anonymous Requests From Specific Network"

Example Policy: Rejecting Unauthenticated Requests

The following example replaces the default policies that allow anonymous access and authenticated access with a single policy for authenticated access. This example then adds a policy to allow anonymous access to use the StartTLS and Get Symmetric Key extended operations. It does not change the default policy that allows anonymous access to read root DSE operational attributes, as that information should remain publicly readable in most deployments:

```
$ dsconfig \
 delete-global-access-control-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Anonymous access all entries" \
 --trustAll \
 --no-prompt
$ dsconfig \
 delete-global-access-control-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Authenticated access all entries" \
 --trustAll \
 --no-prompt
$ dsconfig \
 create-global-access-control-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Authenticated access all entries" \
 --set authentication-required:true \
 --set request-target-dn-not-equal-to:"**,cn=changelog" \
 --set permission:read \
 --set allowed-attribute:"*" \
 --set allowed-attribute:createTimestamp \
 --set allowed-attribute:creatorsName \
 --set allowed-attribute:entryDN \
```

```
--set allowed-attribute:entryUUID \
--set allowed-attribute:etag \
--set allowed-attribute:governingStructureRule \
--set allowed-attribute:hasSubordinates \
--set allowed-attribute:isMemberOf \
--set allowed-attribute:modifiersName \
--set allowed-attribute:modifyTimestamp \
--set allowed-attribute:numSubordinates \
--set allowed-attribute:structuralObjectClass \
--set allowed-attribute:subschemaSubentry \
--set allowed-attribute-exception:authPassword \
--set allowed-attribute-exception:userPassword \
--set allowed-attribute-exception:debugSearchIndex \
--set allowed-attribute-exception:@changeLogEntry \
--set allowed-control:Assertion \
--set allowed-control:AuthorizationIdentity \
--set allowed-control:Csn \
--set allowed-control:ManageDsaIt \
--set allowed-control:MatchedValues \
--set allowed-control:Noop \
--set allowed-control:PasswordPolicy \
--set allowed-control:PermissiveModify \
--set allowed-control:PostRead \
--set allowed-control:PreRead \
--set allowed-control:ProxiedAuthV2 \
--set allowed-control:RealAttributesOnly \
--set allowed-control:ServerSideSort \
--set allowed-control:SimplePagedResults \
--set allowed-control:TransactionId \
--set allowed-control:VirtualAttributesOnly \
--set allowed-control:Vlv \
--set allowed-extended-operation:GetSymmetricKey \
--set allowed-extended-operation:PasswordModify \
--set allowed-extended-operation:PasswordPolicyState \
--set allowed-extended-operation:StartTls \
--set allowed-extended-operation:WhoAmI \
--trustAll \
--no-prompt
$ dsconfig \
create-global-access-control-policy \
--hostname opendj.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--policy-name "Anonymous extended operation access" \
--set authentication-required:false \
--set allowed-extended-operation:GetSymmetricKey \
--set allowed-extended-operation:StartTls \
--trustAll \
--no-prompt
```

Example Policy: Require Secure Connections

The following example creates a policy with a minimum security strength factor of 128, effectively allowing only secure connections for requests targeting data in `dc=example,dc=com`. A security strength factor defines the key strength for DIGEST-MD5, GSSAPI, SSL, and TLS:

```
$ dsconfig \
 create-global-access-control-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Require secure connections for example.com data" \
 --set request-target-dn-equal-to:"**,dc=example,dc=com" \
 --set request-target-dn-equal-to:dc=example,dc=com \
 --set connection-minimum-ssf:128 \
 --trustAll \
 --no-prompt
```

Example Policy: Allow Anonymous Requests From Specific Network

The following example updates policies that allow anonymous requests so they are scoped to apply to clients in the `example.com` domain:

```
$ dsconfig \
 set-global-access-control-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Anonymous extended operation access" \
 --set connection-client-address-equal-to:.example.com \
 --trustAll \
 --no-prompt
$ dsconfig \
 set-global-access-control-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Root DSE access" \
 --set connection-client-address-equal-to:.example.com \
 --trustAll \
 --no-prompt
```

With the `connection-client-address-not-equal-to` property, it is also possible to reject requests from a particular host, domain, address, or address mask.

For additional details, see the reference documentation concerning Global access control policies.

6.7. Viewing Effective Rights

As the number of ACIs increases, it can be difficult to understand by inspection what rights a user actually has to a given entry. The following example shows how the Get Effective Rights control can help.

By default, only users such as Directory Manager who can bypass ACIs can use the Get Effective Rights control (OID `1.3.6.1.4.1.42.2.27.9.5.2`), and related operational attributes, `aclRights` and `aclRightsInfo`. For this example, explicitly grant access to My App using global ACIs:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --add global-aci:\(targetcontrol=\"1.3.6.1.4.1.42.2.27.9.5.2\"\)\ \(version\ 3.0\;acl\ \"Allow\ My\ App\
 to\ get\ effective\ rights\"\;\ allow\(read\)\ userdn=\"ldap:///cn=My\ App,ou=Apps,dc=example,dc=com\"\;
\) \
 --add global-aci:\(targetattr=\"aclRights\|\|aclRightsInfo\"\)\(version\ 3.0\;\ acl\ \"Allow\ My\ App\
 to\ read\ effective\ rights\ attributes\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///cn=My\ App
,ou=Apps,dc=example,dc=com\"\;\) \
 --trustAll \
 --no-prompt
```

In this example, Babs Jensen is the owner of a small group of people who are willing to carpool:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)"
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
description: People who are willing to carpool
cn: Carpoolers
owner: uid=bjensen,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
```

When My App performs the same search with the get effective rights control, and requests the `aclRights` attribute, it sees the rights that it has on the entry:

```
$ ldapsearch \
 --control effectiverights \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:1,delete:1,read:1,write:1,proxy:1
```

When you request the `aclRightsInfo` attribute, the server responds with information about the ACIs applied:

```
$ ldapsearch \
 --control effectiverights \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights \
 aclRightsInfo
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRightsInfo;logs;entryLevel;add: acl_summary(main): access allowed(add) on ent
 ry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL) to (cn
 =My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated allow , de
 ciding_aci: Allow apps proxied auth)
aclRightsInfo;logs;entryLevel;delete: acl_summary(main): access allowed(delete)
 on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL)
 to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated allo
 w , deciding_aci: Allow apps proxied auth)
aclRightsInfo;logs;entryLevel;read: acl_summary(main): access allowed(read) on e
 ntry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, objectClas
 s) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated a
 llow , deciding_aci: Anonymous read-search access)
aclRightsInfo;logs;entryLevel;write: acl_summary(main): access allowed(write) on
  entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL) to
  (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated allow
 , deciding_aci: Allow apps proxied auth)
aclRightsInfo;logs;entryLevel;proxy: acl_summary(main): access allowed(proxy) on
  entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL) to
  (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated allow
 , deciding_aci: Allow apps proxied auth)
aclRights;entryLevel: add:1,delete:1,read:1,write:1,proxy:1
```

You can also request the effective rights for another user by using the `--getEffectiveRightsAuthzid` option (short form: `-g`). This option takes the authorization identity of the user as an argument. The following example shows My App checking Babs's rights to the same entry:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:uid=bjensen,ou=People,dc=example,dc=com" \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:0,delete:1,read:1,write:0,proxy:0
```

The following example checks anonymous user rights to the same entry. Notice that the authorization identity for an anonymous user is expressed as `dn::`:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:" \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

To check access to an attribute that might not yet exist on the entry, use the `--getEffectiveRightsAttribute` option (short form: `-e`). This option takes a comma-separated attribute list as an argument. The following example checks Andy Hall's access to the member attribute for the Carpooler's group entry:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:uid=ahall,ou=People,dc=example,dc=com" \
 --getEffectiveRightsAttribute member \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "cn=Carpoolers" \
 aclRights
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;attributeLevel;member: search:1,read:1,compare:1,write:0,selfwrite_add:1,selfwrite_delete:1
,proxy:0
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

Chapter 7
Indexing Attribute Values

This chapter covers OpenDJ indexing features used to speed up searches, and to limit the impact of searches on directory server resources. In this chapter you will learn to:

• Define indexes and explain why they are useful

• Determine what to index and what types of indexes to use

• Configure, build, and rebuild indexes

• Check that indexes are valid

7.1. About Indexes

A basic, standard directory feature is the ability to respond quickly to searches.

An LDAP search specifies the following information that directly affects how long the directory might take to respond:

• The base DN for the search.

  The more specific the base DN, the less information to check during the search. For example, a request with base DN `dc=example,dc=com` potentially involves checking many more entries than a request with base DN `uid=bjensen,ou=people,dc=example,dc=com`.

• The scope of the search.

  A subtree or one-level scope targets many entries, whereas a base search is limited to one entry.

• The search filter to match.

  A search filter, such as `(cn=Babs Jensen)`, asserts that an attribute on the entry to search for, in this case `cn`, corresponds to some value. In this case, the attribute must have a value that equals `Babs Jensen`, ignoring case sensitivity.

  It would generally be a waste of resources to have the directory server check all entries to see whether they have a CN of Babs Jensen. Instead, directory servers maintain indexes to expedite checking whether a search filter matches.

LDAP directory servers like OpenDJ directory server even go so far as to disallow searches that cannot be handled expediently using indexes. Maintaining appropriate indexes is a key aspect of directory administration.

The role of an index is to answer the question, "Which entries have an attribute with this corresponding value?" Each index is therefore specific to an attribute. Each index is also specific to the comparison implied in the search filter. For example, an OpenDJ directory server maintains distinct indexes for exact (equality) matching and for substring matching. The types of indexes are explained in "Index Types and Their Functions". Furthermore, indexes are configured in specific directory backends.

An OpenDJ index is implemented as a tree of key-value pairs. The key is a form of the value to match, such as `babs jensen`. The value is a list of IDs for entries that match the key. "An OpenDJ Equality Index" shows an equality (case ignore exact match) index with five keys from a total of four entries. If the data set were large, there could be more than one entry ID per key.

An OpenDJ Equality Index
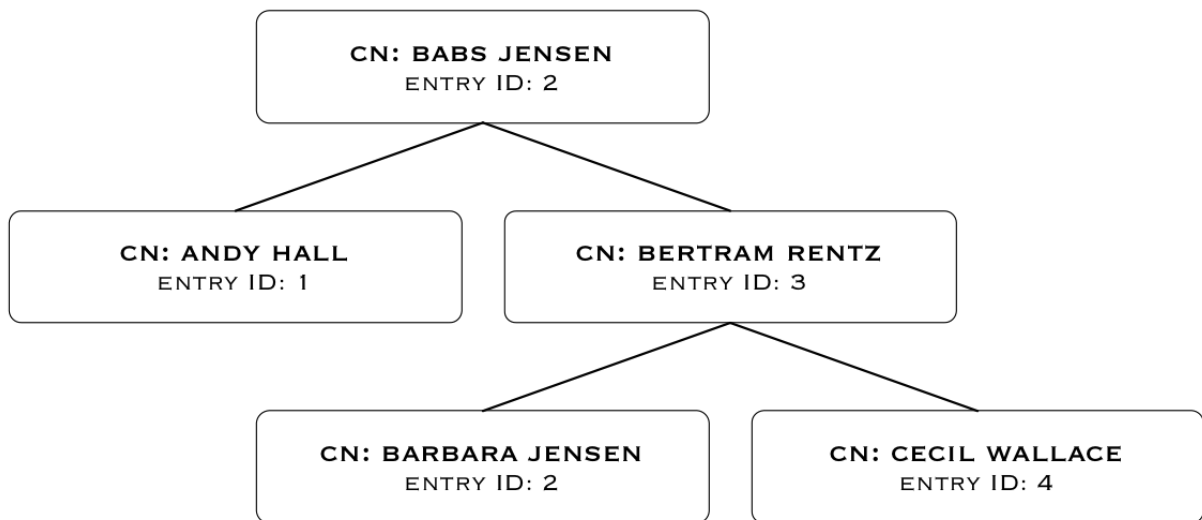
This is how an OpenDJ directory server uses indexes. When the search filter is `(cn=Babs Jensen)`, the OpenDJ directory server retrieves the IDs for entries with a CN matching `Babs Jensen` from the equality index of the CN attribute. (For a complex filter, the OpenDJ directory server might optimize the search by changing the order in which it uses the indexes.) A successful result is zero or more entry IDs. These are the candidate result entries.

For each candidate, the OpenDJ directory server retrieves the entry by ID from a special system index called `id2entry`, which, as its name suggests, returns an entry for an entry ID. If there is a match, and

the client application has the right to access to the data, the OpenDJ directory server returns the search result. It continues this process until no candidates are left.

If there are no indexes that correspond to a search request, then the OpenDJ directory server must potentially check for a match against every entry in the scope of the search. Evaluating every entry for a match is referred to as an *unindexed* search. An unindexed search is an expensive operation, particularly for large directories. For this reason, an OpenDJ directory server refuses unindexed searches unless the user making the request has specific permission to make such requests. Permission to perform an unindexed search is granted with the `unindexed-search` privilege. This privilege is reserved for the directory root user by default, and should not be granted lightly.

If the number of user data entries is smaller than the default resource limits, you can still perform what appear to be unindexed searches, meaning searches with filters for which no index appears to exist. That is because the `dn2id` index returns all user data entries without hitting a resource limit that would make the search unindexed.

7.2. What To Index

OpenDJ search performance depends on indexes as described in "About Indexes".

An OpenDJ directory server maintains generally useful indexes for data imported into the default `userRoot` backend. When you create a new backend, the OpenDJ directory server only maintains the necessary system indexes unless you configure additional indexes. For details, see "Default Indexes".

The default settings are fine for evaluating OpenDJ software, and they work well with sample data. The default settings might not, however, fit your directory data and the searches performed on your directory service.

You can view and edit what is indexed through OpenDJ control panel, Indexes > Manage Indexes. Alternatively, you can manage indexes using the command-line tools demonstrated in "Configuring and Rebuilding Indexes".

7.2.1. Determining Which Indexes Are Needed

Index maintenance has its costs. Every time an indexed attribute is updated, the OpenDJ directory server must update each affected index to reflect the change, which is wasteful if the index is hardly used. Indexes, especially substring indexes, can take up more memory and disk space than the corresponding data.

Aim to maintain only those indexes that speed up appropriate searches, and that allow the OpenDJ directory server to operate properly. The latter indexes include non-configurable internal indexes, and generally are handled by the OpenDJ directory server without intervention. The former, indexes for appropriate searches, require thought and investigation. Whether a search is appropriate depends on the circumstances.

Begin by reviewing the attributes of your directory data. Which attributes would you expect to see in a search filter? If an attribute is going to show up frequently in reasonable search filters, then it ought to be indexed.

Compare your guesses with what you see actually happening in the directory. One way of doing this is to review the access log for search results that are marked with additional items including `unindexed`:

```
$ grep unindexed /path/to/opendj/logs/ldap-access.audit.json
{...,"request":{"protocol":"LDAP","operation":"SEARCH",...,"dn":"ou=people,dc=example,dc=com"
,"scope":"sub","filter":"(&(mail=*.com)(objectclass=person))","attrs":["ALL"]},..,"response":
{"status":"FAILED","statusCode":"50",...,"detail":"You do not have sufficient privileges to perform an
 unindexed search","additionalItems":" unindexed","nentries":0},...}
{...,"request":{"protocol":"LDAP","operation":"SEARCH",...,"dn":"ou=people,dc=example,dc=com"
,"scope":"sub","filter":"(&(employeenumber=86182)(mail=*@example.com))","attrs":["ALL"]},...,"response":
{"status":"FAILED","statusCode":"50",...,"detail":"You do not have sufficient privileges to perform an
 unindexed search","additionalItems":" unindexed","nentries":0},...}
{...,"request":{"protocol":"LDAP","operation":"SEARCH",...,"dn":"ou=people,dc=example,dc=com"
,"scope":"sub","filter":"(objectclass=person)","attrs":["ALL"]},...,"response":{"status":"FAILED"
,"statusCode":"50",...,"detail":"You do not have sufficient privileges to perform an unindexed search"
,"additionalItems":" unindexed","nentries":0},...}
```

Understand the search filter that led to each unindexed search. If the filter is appropriate and frequently used, add an index to facilitate the search. You can either consume the access logs to determine how often a search filter is used, or monitor what is happening in the directory by using the index analysis feature.

OpenDJ servers provide this feature to collect information about filters in search requests. You can activate the index analysis mechanism using the **dsconfig set-backend-prop** command:

```
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set index-filter-analyzer-enabled:true \
 --no-prompt \
 --trustAll
```

The command causes the server to analyze filters used, and to keep the results in memory, so that you can read them through the `cn=monitor` interface:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "cn=userRoot Storage,cn=monitor" \
 "(objectclass=*)" \
 filter-use
dn: cn=userRoot Storage,cn=monitor
filter-use: (sn=Ishee) hits:1 maxmatches:8 message:
filter-use: (mail=*.com) hits:1 maxmatches:-1 message:The filter value exceeded the index entry limit for
 the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 index
filter-use: (uid=user.86182) hits:1 maxmatches:1 message:
filter-use: (objectclass=person) hits:2 maxmatches:-1 message:The filter value exceeded the index entry
 limit for the /dc=com,dc=example/objectClass.objectIdentifierMatch index
filter-use: (employeenumber=86182) hits:1 maxmatches:-1 message:equality index type is disabled for the
 employeeNumber attribute
filter-use: (mail=*@example.com) hits:1 maxmatches:-1 message:The filter value exceeded the index entry
 limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter value exceeded
 the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter
 value exceeded the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6
 indexThe filter value exceeded the index entry limit for the /dc=com,dc=example/mail
.caseIgnoreIA5SubstringsMatch:6 indexThe filter value exceeded the index entry limit for the /dc=com
,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter value exceeded the index entry limit
 for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter value exceeded the index
 entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter value exceeded
 the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6 indexThe filter
 value exceeded the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6
 indexThe filter value exceeded the index entry limit for the /dc=com,dc=example/mail
.caseIgnoreIA5SubstringsMatch:6 index
```

The `filter-use` values are the filter, the `hits` (number of times the filter was used), the `maxmatches` (number of matches found), and an optional message.

The output can include filters for internal use, such as `(aci=*)`. In the example above, you see filters for client application searches.

One appropriate search filter that led to an unindexed search, `(employeenumber=86182)`, had no matches because, "equality index type is disabled for the employeeNumber attribute." Some client application is trying to find specific users by employee number, but no index exists for that purpose. If this appears regularly as a frequent search, add an employee number index as described in "Configuring a Standard Index".

One inappropriate search filter that led to an unindexed search, `(mail=*.com)`, had no matches because, "The filter value exceeded the index entry limit for the /dc=com,dc=example/ mail.caseIgnoreIA5SubstringsMatch:6 index." It appears that some client application is trying to list all entries with an email address ending in `.com`. There are so many such entries that although an index exists for the `mail` attribute, the server has given up maintaining the list of entries with email addresses ending in `.com`. In a large directory, there might be many thousands of matching entries. If you take action to allow this expensive search, the requests could consume a large share of directory resources, or even cause a denial of service to other requests.

To avoid impacting server performance, turn off index analysis after you collect the information you need. You turn off index analysis with the **dsconfig set-backend-prop** command:

```
$ dsconfig \
  set-backend-prop \
  --hostname opendj.example.com \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --backend-name userRoot \
  --set index-filter-analyzer-enabled:false \
  --no-prompt \
  --trustAll
```

Directory users might complain to you that their searches are refused because they are unindexed. Ask for the result code, additional information, and search filter. An OpenDJ directory server responds to an LDAP client application that attempts an unindexed search with a result code of 50 and additional information about an unindexed search. The following example attempts, anonymously, to get the entries for all users whose email address ends in `.com`:

```
$ ldapsearch \
  --port 1389 \
  --baseDN ou=people,dc=example,dc=com \
  "(&(mail=*.com)(objectclass=person))"
# The LDAP search request failed: 50 (Insufficient Access Rights)
# Additional Information:  You do not have sufficient privileges to perform an unindexed search
```

Rather than adjusting settings to permit the search, try to understand why the user wants to perform an unindexed search.

Perhaps they are unintentionally requesting an unindexed search. If so, you can help them find a less expensive search, by using an approach that limits the number of candidate result entries. For example, if a GUI application lets a user browse a group of entries, the application could use a browsing index to retrieve a block of entries for each screen, rather than retrieving all the entries at once.

Perhaps they do have a legitimate reason to get the full list of all entries in one operation, such as regularly rebuilding some database that depends on the directory. If so, their application can perform the search as a user who has the `unindexed-search` privilege. To assign the `unindexed-search` privilege, see "Configuring Privileges".

7.2.2. Clarifying Which Indexes Are Used by a Search

Sometimes it is not obvious by inspection how an OpenDJ directory server handles a given search request internally. The directory root user can inspect how an OpenDJ directory server resolves the search request by performing the same search with the `debugsearchindex` attribute.

A default global access control setting prevents users from reading the `debugsearchindex` attribute. To allow an administrator to read the attribute, add a global access control setting as in the following example for a directory server using ACIs:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --add global-aci:"(targetattr=\"debugsearchindex\")(version 3.0; acl \"Debug search indexes\"; \
  allow (read,search,compare) userdn=\"ldap:///uid=user.0,ou=people,dc=example,dc=com\";)" \
 --trustAll \
 --no-prompt
```

The following example demonstrates this feature for an exact match search:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=user.0,ou=people,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=user.1000)" \
 debugsearchindex
dn: cn=debugsearch
debugsearchindex: filter=(uid=user.1000)[INDEX:uid.equality][COUNT:1] final=[COUNT:1]
```

When you request the `debugsearchindex` attribute, instead of performing the search, the OpenDJ directory server returns debug information indicating how it would process the search operation. In the example above, notice that the OpenDJ directory server uses the equality index for the `uid` attribute.

A search with a less exact filter requires more work. In the following example the search filter would match thousands of entries:

```
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN uid=user.0,ou=people,dc=example,dc=com \
 --bindPassword password \
 "(uid=*)" \
 debugsearchindex
dn: cn=debugsearch
debugsearchindex: filter=(uid=*)[INDEX:uid.presence][NOT-INDEXED] scope=sub[NOT-INDEXED] final=[NOT-
INDEXED]
```

Although an index exists, the set of results is so large that the server has stopped maintaining the list of entry IDs, and so the search is considered unindexed.

If an index already exists, but you suspect it is not working properly, see "Verifying Indexes".

About `debugsearchindex` Values

The values of the `debugsearchindex` attribute show you how an OpenDJ directory server uses search filters and scope to determine the results of the search. In general, the `debugsearchindex` attribute has the form: `(filter|vlv)=`*filter-with-info*`( scope=`*scope-id**scope-info*`) final=`*final-info*.

If a normal filter applies, the value starts with `filter=`. If the search operation parameters have an associated VLV index, the value starts with `vlv=`. A `scope` component provides information about how the scope affected the results. The `final` component provides information about the overall result.

filter-with-info

This field looks like a string representation of the LDAP filter with extra information after the closing parenthesis of each simple filter component.

For a VLV index, only the extra information is shown:

The extra information takes the form: `([INDEX:`*index-id*`])([COUNT:`*entry-count*`]|[LIMIT-EXCEEDED]|[NOT-INDEXED])`, where:

- `[INDEX:`*index-id*`]` identifies the index that could be used to find matches for this filter.

- `[COUNT:`*entry-count*`]` specifies the number of entries found to match the filter.

- `[LIMIT-EXCEEDED]` indicates the server maintains a matching index, but the index entry limit was exceeded for the value specified.

- `[NOT-INDEXED]` indicates no matching index value or index key was found.

For example, the `debugsearchindex` attribute value excerpt `filter=(&(objectClass=person) [INDEX:objectClass.equality] [LIMIT-EXCEEDED](cn=*a*)[INDEX:cn.substring][NOT-INDEXED])[NOT-INDEXED]` provides information about how OpenDJ evaluates the complex filter `(&(objectClass=person) (cn=*a*))`. The filter component `(objectClass=person)` does correspond to the equality index for `objectClass`, but there are so many entries matching `objectClass=person` that the server has stopped maintaining index entries for that value. The filter component `cn=*a*` did not match an index, as might be expected for such a short substring. No matching index was found for the whole complex filter.

scope-id

The scope can be one of `base`, `one`, `sub`, or `subordinate`.

scope-info

This field is similar to the extra information for filter components:

- `[COUNT:`*entry-count*`]` specifies the number of entries found in the scope.

- `[LIMIT-EXCEEDED:entry-count]` indicates the scope did not prevent the search from exceeding the resource limit that caps the number of entries a search can return.

For example, the `debugsearchindex` attribute value excerpt `scope=sub[LIMIT-EXCEEDED:10002]` indicates that the number of matches in the subtree scope that exceeded the resource limit capping how many entries a search can return.

final-info

This field shows at a glance whether the search was indexed:

- `[COUNT:entry-count]` specifies the number of entries found, and indicates that the search was indexed.

- `[NOT-INDEXED]` indicates that the search was unindexed.

7.3. Index Types and Their Functions

OpenDJ directory servers support multiple index types, each corresponding to a different type of search. This section describes the index types and what they are used for.

View what is indexed through OpenDJ control panel, Indexes > Manage Indexes. Alternatively, use the **backendstat list-indexes** command. For details about a particular index, you can use the **backendstat dump-index** command.

7.3.1. Presence Index

A presence index is used to match an attribute that is present on the entry, regardless of the value. The `aci` attribute is indexed for presence by default to allow quick retrieval of entries with ACIs:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(aci=*)" -
```

Due to its implementation, a presence index takes up less space than other indexes. In a presence index, there is just one key with a list of IDs.

As described in "About Indexes", an OpenDJ index is implemented as a tree of key-value pairs. The following command examines the ACI presence index for Example.ldif data:

```
$ stop-ds --quiet
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName aci.presence
Key (len 1): PRESENCE
Value (len 6): [COUNT:3] 100003 100011 100013

Total Records: 1
Total / Average Key Size: 1 bytes / 1 bytes
Total / Average Data Size: 6 bytes / 6 bytes
```

In this case, entries with ACI attributes have IDs `100003`, `100011`, and `100013`.

7.3.2. Equality Index

An equality index is used to match values that correspond exactly (though generally without case sensitivity) to the value provided in the search filter. An equality index requires clients to match values without wildcards or misspellings:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)" mail
dn: uid=bjensen,ou=People,dc=example,dc=com
mail: bjensen@example.com
```

An equality index has one list of entry IDs for each attribute value. Depending on the backend implementation, the keys in a case-insensitive index might not be strings. For example, a key of `6A656E73656E` could represent `jensen`.

As described in "About Indexes", an OpenDJ index is implemented as a tree of key-value pairs. The following command examines the SN equality index for Example.ldif data:

```
$ stop-ds --quiet
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreMatch
...
Key (len 6): jensen
Value (len 12): [COUNT:9] 100020 100033 100034 100068 100081 100096 100135 100136 100152
...
```

In this case, there are nine entries that have an SN of Jensen.

As long as the keys of the equality index are not encrypted, an OpenDJ directory server can reuse an equality index for some other searches, such as ordering and initial substring searches.

7.3.3. Approximate Index

An approximate index is used to match values that "sound like" those provided in the filter. An approximate index on `cn` lets client applications find people even when they misspell names, as in the following example:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(cn~=Babs Jansen)" cn
dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

An approximate index squashes attribute values into a normalized form.

As described in "About Indexes", an OpenDJ index is implemented as a tree of key-value pairs. The following command examines an SN approximate index for Example.ldif data:

```
$ stop-ds --quiet
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName sn.ds-mr-double-metaphone-approx
...
Key (len 4): JNSN
Value (len 13): [COUNT:10] 100020 100033 100034 100061 100068 100081 100096 100135 100136 100152
...
```

In this case, there are ten entries that have an SN that sounds like Jensen.

7.3.4. Substring Index

A substring index is used to match values that are specified with wildcards in the filter. Substring indexes can be expensive to maintain, especially for large attribute values:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(cn=Barb*)" cn
dn: uid=bfrancis,ou=People,dc=example,dc=com
cn: Barbara Francis

dn: uid=bhal2,ou=People,dc=example,dc=com
cn: Barbara Hall

dn: uid=bjablons,ou=People,dc=example,dc=com
cn: Barbara Jablonski

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen

dn: uid=bmaddox,ou=People,dc=example,dc=com
cn: Barbara Maddox
```

In a substring index, there are enough keys to allow the server to match any substring in the attribute values. Each key is associated with a list of IDs. The default maximum size of a substring key is 6 bytes.

As described in "About Indexes", an OpenDJ index is implemented as a tree of key-value pairs. The following command examines an SN substring index for Example.ldif data:

```
$ stop-ds --quiet
$ backendstat \
dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreSubstringsMatch:6
...
Key (len 1): e
Value (len 25): [COUNT:22] ..
.
...
Key (len 2): en
Value (len 15): [COUNT:12] ..
.
...
Key (len 3): ens
Value (len 4): [COUNT:1] 100149
Key (len 5): ensen
Value (len 12): [COUNT:9] 100020 100033 100034 100068 100081 100096 100135 100136 100152
...
Key (len 6): jensen
Value (len 12): [COUNT:9] 100020 100033 100034 100068 100081 100096 100135 100136 100152
...
Key (len 1): n
Value (len 35): [COUNT:32] ..
.
...
Key (len 2): ns
Value (len 4): [COUNT:1] 100149
Key (len 4): nsen
Value (len 12): [COUNT:9] 100020 100033 100034 100068 100081 100096 100135 100136 100152
...
Key (len 1): s
Value (len 15): [COUNT:12] 100014 100028 100049 100066 100097 100100 100110 100133 100137 100149 100151
  100156
...
Key (len 2): se
Value (len 9): [COUNT:6] 100054 100060 100077 100119 100125 100150
Key (len 3): sen
Value (len 12): [COUNT:9] 100020 100033 100034 100068 100081 100096 100135 100136 100152
...
```

In this case, the SN value Jensen shares substrings with many other entries. Given the size of the lists and number of keys in a substring index, it is much more expensive to maintain than other indexes. This is particularly true for longer attribute values.

7.3.5. Ordering Index

An ordering index is used to match values for a filter that specifies a range. For example, the `ds-sync -hist` attribute, which is for the server's internal use, has an ordering index by default. Searches on that attribute often seek entries with changes more recent than the last time a search was performed.

The following example shows a search that specifies a range on the SN attribute value:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(sn>=winter)" sn
dn: uid=aworrell,ou=People,dc=example,dc=com
sn: Worrell

dn: uid=kwinters,ou=People,dc=example,dc=com
sn: Winters

dn: uid=pworrell,ou=People,dc=example,dc=com
sn: Worrell
```

In this case, the server only requires an ordering index if it cannot reuse the (ordered) equality index instead. For example, if the equality index is encrypted, the ordering index would need to be maintained separately.

7.3.6. Virtual List View (Browsing) Index

A virtual list view (VLV) or browsing index is designed to help the server respond to client applications that need virtual list view results, for example, to browse through a long list in a GUI. They also help the server respond to clients that request server-side sorting of the search results.

VLV indexes correspond to particular searches. Configure your VLV indexes using the control panel, and copy the command-line equivalent from the Details pane for the operation, if necessary.

7.3.7. Extensible Matching Rule Index

In some cases you need an index for a matching rule other than those described above. For example, OpenDJ supports generalized time-based matching so that applications can search for all times later than, or earlier than a specified time.

7.4. Configuring and Rebuilding Indexes

You modify index configurations by using the **dsconfig** command. The subcommands to use depend on the backend type, as shown in the examples that follow. The configuration changes then take effect after you rebuild the index according to the new configuration, using the **rebuild-index** command. The **dsconfig --help-database** command lists subcommands for creating, reading, updating, and deleting index configuration.

> **Tip**
>
> Indexes are per directory backend rather than per suffix. To maintain separate indexes for different suffixes on the same directory server, put the suffixes in different backends.

This section includes the following procedures:

- "Configuring a Standard Index"

- "Configuring a Virtual List View Index"

- "Rebuilding Indexes"

- "Understanding Index Entry Limits"

7.4.1. Configuring a Standard Index

You can configure standard indexes from the control panel, and also on the command-line using the **dsconfig** command. After you finish configuring the index, you must rebuild the index for the changes to take effect.

To prevent indexed values from appearing in cleartext in a backend, you can enable confidentiality by backend index. For details, see "Encrypting Directory Data".

This section includes the following examples:

- "Create a New Index"

- "Add an Approximate Index"

- "Configure an Extensible Match Index"

- "Configuring an Index for a JSON Attribute"

Create a New Index

The following example creates a new equality index for the `description` attribute in the default `userRoot` backend:

```
$ dsconfig \
 create-backend-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name description \
 --set index-type:equality \
 --trustAll \
 --no-prompt
```

Add an Approximate Index

The following example add an approximate index for the `sn` (surname) attribute in the default `userRoot` backend:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name sn \
 --add index-type:approximate \
 --trustAll \
 --no-prompt
```

Approximate indexes depend on the Double Metaphone matching rule, described in "Configure an Extensible Match Index".

Configure an Extensible Match Index

OpenDJ servers support matching rules defined in LDAP RFCs. They also define OpenDJ-specific extensible matching rules.

The following are OpenDJ-specific extensible matching rules:

Name: `ds-mr-double-metaphone-approx`
OID: `1.3.6.1.4.1.26027.1.4.1`

Double Metaphone Approximate Match described at http://aspell.net/metaphone/. The OpenDJ implementation always produces a single value rather than one or possibly two values.

Configure approximate indexes as described in "Add an Approximate Index".

For an example using this matching rule, see "Search: Finding an Approximate Match" in the *Developer's Guide*.

Name: `ds-mr-user-password-exact`
OID: `1.3.6.1.4.1.26027.1.4.2`

User password exact matching rule used to compare encoded bytes of two hashed password values for exact equality.

Name: `ds-mr-user-password-equality`
OID: `1.3.6.1.4.1.26027.1.4.3`

User password matching rule implemented as the user password exact matching rule.

Name: `partialDateAndTimeMatchingRule`
OID: `1.3.6.1.4.1.26027.1.4.7`

Partial date and time matching rule for matching parts of dates in time-based searches.

For an example using this matching rule, see "Search: Listing Active Accounts" in the *Developer's Guide*.

Name: `relativeTimeOrderingMatch.gt`
OID: `1.3.6.1.4.1.26027.1.4.5`

Greater-than relative time matching rule for time-based searches.

For an example that configures an index with this matching rule, see "Search: Listing Active Accounts" in the *Developer's Guide*.

Name: `relativeTimeOrderingMatch.lt`
OID: `1.3.6.1.4.1.26027.1.4.6`

Less-than relative time matching rule for time-based searches.

For an example using this matching rule, see "Search: Listing Active Accounts" in the *Developer's Guide*.

The OpenDJ control panel New Index window does not help you set up extensible matching rule indexes. Use the **dsconfig** command instead.

The following example configures an extensible matching rule index for "later than" and "earlier than" generalized time matching on a `lastLoginTime` attribute in the default `userRoot` backend:

```
$ dsconfig \
 create-backend-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set index-type:extensible \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.5 \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.6 \
 --index-name lastLoginTime \
 --trustAll \
 --no-prompt
```

Configuring an Index for a JSON Attribute

OpenDJ servers support attribute values that have JSON syntax. This makes it possible to index JSON values and to search for them using Common REST query filters, as demonstrated in "Search: Using JSON Query Filters" in the *Developer's Guide*.

A JSON attribute has `Json` syntax (OID: `1.3.6.1.4.1.36733.2.1.3.1`). It uses either a case-sensitive or case-insensitive `Json Query` matching rule. The following schema definition, shown in `Example.ldif`, defines a `json` attribute with case-insensitive matching:

```
attributeTypes: ( json-attribute-oid NAME 'json'
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1 EQUALITY caseIgnoreJsonQueryMatch
  SINGLE-VALUE X-ORIGIN 'OpenDJ Documentation Examples' )
```

By default, an equality index for a JSON syntax attribute indexes all JSON fields in each JSON object. The following command creates an equality index for a `json` attribute:

```
$ dsconfig \
 create-backend-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name json \
 --set index-type:equality \
 --trustAll \
 --no-prompt
```

The server uses the equality index for all query filter matching, not just equality. If you encrypt the index as described in "Encrypting Directory Data" however, then it supports only equality matching.

Large JSON objects stored on JSON attributes can cause the large indexes, because the default behavior for `Json Query` matching rules is to maintain index keys for each JSON field. If you know the JSON fields client applications will use in query filters, you can change that behavior. You do this by configuring a custom schema provider for `Json Query` the matching rule used by the JSON attribute to index.

The `Json Query` matching rules are the following:

caseIgnoreJsonQueryMatch

Ignore case when finding matches. For example, `Babs` and `babs` match.

OID: `1.3.6.1.4.1.36733.2.1.4.1`

caseExactJsonQueryMatch

Respect case when finding matches. For example, `Babs` and `babs` *do not* match.

OID: `1.3.6.1.4.1.36733.2.1.4.2`

The following example configures a custom schema provider for the case-insensitive JSON matching rule that only maintains keys for the `access_token` field, and all fields beneath `contact_information`:

```
$ dsconfig \
 create-schema-provider \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Json Schema" \
 --type json-schema \
 --set enabled:true \
 --set case-sensitive-strings:false \
 --set ignore-white-space:true \
 --set matching-rule-name:caseIgnoreJsonQueryMatch \
 --set matching-rule-oid:1.3.6.1.4.1.36733.2.1.4.1 \
 --set indexed-field:access_token \
 --set "indexed-field:contact_information/**" \
 --trustAll \
 --no-prompt
```

After you rebuild the index, as described in "Rebuilding Indexes", the change takes effect. Bear in mind that this impacts indexes for all JSON attributes using the matching rule.

7.4.2. Configuring a Virtual List View Index

In the OpenDJ control panel, select Manage Indexes > New VLV Index, and then set up your VLV index using the New VLV Index window as shown in "New VLV Index Window".

New VLV Index Window

After you finish configuring your index and click OK, the Control Panel prompts you to make the additional changes necessary to complete the VLV index configuration, and then to build the index.

You can also create the equivalent index configuration by using the **dsconfig** command.

The following example shows how to create the VLV index for the default userRoot backend:

```
$ dsconfig \
 create-backend-vlv-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name people-by-last-name \
 --set base-dn:ou=People,dc=example,dc=com \
 --set filter:"(|(givenName=*)(sn=*))" \
 --set scope:single-level \
 --set sort-order:"+sn +givenName" \
 --trustAll \
 --no-prompt
```

Note

When referring to a VLV index after creation, you must add `vlv.` as a prefix. In other words, if you named the VLV index `people-by-last-name`, you refer to it as `vlv.people-by-last-name` when rebuilding indexes, changing index properties such as the index entry limit, or verifying indexes.

7.4.3. Rebuilding Indexes

After you change an index configuration, or when you find that an index is corrupt, you can rebuild the index. When you rebuild indexes, you specify the base DN of the data to index, and either the list of indexes to rebuild or `--rebuildAll`. You can rebuild indexes while the server is offline, or while the server is online. If you rebuild the index while the server is online, then you must schedule the rebuild process as a task.

This section includes the following examples:

- "Rebuild Index"

- "Rebuild Degraded Indexes"

- "Clear New, Unused, Degraded Indexes"

Rebuild Index

The following example rebuilds the `cn` index immediately with the server online:

```
$ rebuild-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --index cn \
 --start 0 \
 --trustAll
```

Rebuild Degraded Indexes

The following example rebuilds degraded indexes immediately with the server online:

```
$ rebuild-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --rebuildDegraded \
 --start 0 \
 --trustAll
```

Clear New, Unused, Degraded Indexes

When you add a new attribute as described in "Updating Directory Schema", and then create indexes for the new attribute, the new indexes appear as degraded, even though the attribute has not yet been used, and so indexes are sure to be empty, rather than degraded.

In this special case, you can safely use the **rebuild-index --clearDegradedState** command to avoid having to scan the entire directory backend before rebuilding the new, unused index. In this example, an index has just been created for newUnusedAttribute.

Before using the **rebuild-index** command, test the index status to make sure that the index has not yet been used: by using the **backendstat** command, described in backendstat(1) in the *Reference*.

OpenDJ servers must be stopped before you use the **backendstat** command:

```
$ stop-ds --quiet
```

The third column of the output is the Valid column, which is false before the rebuild:

```
$ backendstat \
 show-index-status \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 | grep newUnusedAttribute
newUnusedAttribute.presence                 ... false ...
newUnusedAttribute.caseIgnoreMatch          ... false ...
newUnusedAttribute.caseIgnoreSubstringsMatch:6 ... false ...
```

Update the index information to fix the value of the unused index:

```
$ rebuild-index \
 --offline \
 --baseDN dc=example,dc=com \
 --clearDegradedState \
 --index newUnusedAttribute
```

Check that the `Index Valid` column for the index status is now set to `true`:

```
$ backendstat \
 show-index-status \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 | grep newUnusedAttribute
newUnusedAttribute.presence                 ... true ...
newUnusedAttribute.caseIgnoreMatch          ... true ...
newUnusedAttribute.caseIgnoreSubstringsMatch:6 ... true ...
```

Start the server:

```
$ start-ds --quiet
```

If the newly indexed attribute has already been used, rebuild the index instead of clearing the degraded state.

7.4.4. Understanding Index Entry Limits

As described in "About Indexes", an OpenDJ index is implemented as a tree of key-value pairs. The key is what the search is trying to match. The value is a list of entry IDs.

As the number of entries in the directory grows, the list of entry IDs for some keys can become very large. For example, every entry in the directory has the value `top` for the `objectClass` attribute. If the directory maintains a substring index for `mail`, the number of entries ending in `.com` could be huge.

An OpenDJ directory server therefore defines an *index entry limit*. When the number of entry IDs for a key exceeds the limit, the server stops maintaining a list of IDs for that key. The limit effectively

makes a search using that key unindexed. Searches using other keys in the same index are not affected.

"Index Entry Limit Exceeded For a Single Key" shows a fragment from a substring index for the `mail` attribute. The number of email addresses ending in `.com` has exceeded the index entry limit. For the other substring keys, the entry ID lists are still maintained, but to save space the entry IDs are not shown in the diagram.

Index Entry Limit Exceeded For a Single Key
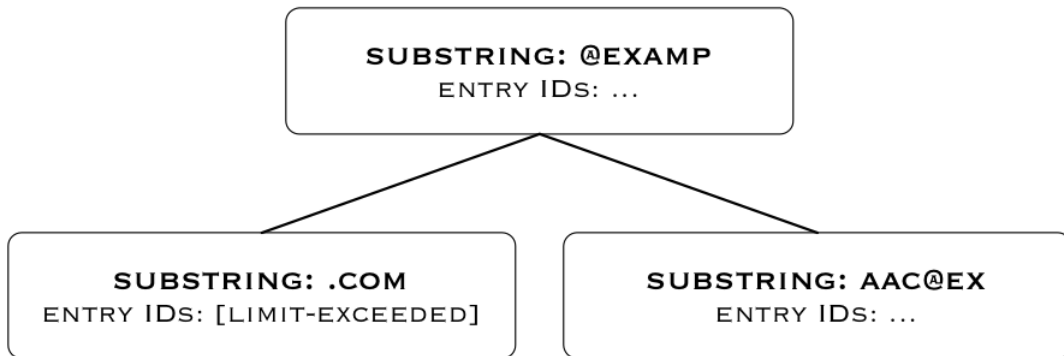
Ideally, the limit is set at the point where it becomes more expensive to maintain the entry ID list for a key and to perform an indexed search than to perform an unindexed search. In practice, the limit is a trade off, with a default index entry limit value of 4000.

To Check Index Entry Limits

The following steps show how to get information about indexes where the index entry limit is exceeded for some keys. In this case, the directory server holds 100,000 user entries. The settings for this directory server are reasonable.

Use the **backendstat show-index-status** command, described in backendstat(1) in the *Reference*.

1. Stop OpenDJ servers before you use the **backendstat** command:

    ```
    $ stop-ds --quiet
    ```

2. Non-zero values in the Over Entry Limit column of the output table indicate the number of keys for which the limit has been reached. The keys that are over the limit are then listed below the table:

```
$ backendstat show-index-status --backendID userRoot --baseDN dc=example,dc=com
Index Name                                  ... Valid  ... Over Entry Limit ..
.
...
mail.caseIgnoreIA5SubstringsMatch:6         ... true   ... 34 ..
.
...
givenName.caseIgnoreSubstringsMatch:6       ... true   ... 9  ...
telephoneNumber.telephoneNumberSubstringsMatch:6 ... true   ... 10 ..
.
...
sn.caseIgnoreSubstringsMatch:6              ... true   ... 14 ..
.
...
cn.caseIgnoreSubstringsMatch:6             ... true   ... 14 ...
objectClass.objectIdentifierMatch          ... true   ... 4  ..
.
...

Index: /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6
Over index-entry-limit keys: [.com] [0@mail] ...

Index: /dc=com,dc=example/givenName.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [e] [i] [ie] [l] [n] [na] [ne] [y]

Index: /dc=com,dc=example/telephoneNumber.telephoneNumberSubstringsMatch:6
Over index-entry-limit keys: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

Index: /dc=com,dc=example/cn.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [an] [e] [er] [i] [k] [l] [n] [o] [on] [r] [s] [t] [y]

Index: /dc=com,dc=example/objectClass.objectIdentifierMatch
Over index-entry-limit keys: [inetorgperson] [organizationalperson] [person] [top]

Index: /dc=com,dc=example/sn.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [an] [e] [er] [i] [k] [l] [n] [o] [on] [r] [s] [t] [y]
```

For example, every user entry has the object classes listed, and every user entry has an email address ending in `.com`, so those values are not specific enough to be used in search filters.

3. Start the server:

```
$ start-ds --quiet
```

Index Entry Limit Changes

In rare cases, the index entry limit might be too low for a certain key. This could manifest itself as a frequent, useful search becoming unindexed, with no reasonable way to narrow the search.

You can change the index entry limit on a per-index basis. Do not do this in production unless you can explain and show why the benefits outweigh the costs.

> **Important**
>
> Changing the index entry limit significantly can result in serious performance degradation. Be prepared to test performance thoroughly before you roll out an index entry limit change in production.

Consider a directory with more than 4000 groups in a backend. When the backend is brought online, an OpenDJ directory server searches for the groups with a search filter of `(|(objectClass=groupOfNames)(objectClass=groupOfEntries)(objectClass=groupOfUniqueNames))`, which is an unindexed search due to the default index entry limit setting. The following example raises the index entry limit for the `objectClass` index to `10000`, and then rebuilds the index for the configuration change to take effect. The steps are the same for any other index:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name objectClass \
 --set index-entry-limit:10000 \
 --no-prompt \
 --trustAll

$ rebuild-index \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --index objectClass \
 --start 0  \
 --trustAll
```

It is also possible, but not recommended, to configure the global `index-entry-limit` for a backend. This changes the default for all indexes in the backend. Use the **dsconfig set-backend-prop** command as shown in the following example:

```
# Not recommended
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set index-entry-limit:10000 \
 --trustAll \
 --no-prompt
```

7.5. Verifying Indexes

You can verify that indexes correspond to current directory data, and that indexes do not contain errors by using the **verify-index** command, described in verify-index(1) in the *Reference*.

Verify Index

The following example verifies the `cn` index offline for completeness and for errors:

```
$ stop-ds --quiet
$ verify-index \
 --baseDN dc=example,dc=com \
 --index cn \
 --clean \
 --countErrors
```

The output indicates whether any errors are found in the index.

7.6. Default Indexes

When you first install an OpenDJ directory server and import your data from LDIF, the indexes are configured as shown in "Default Indexes".

Default Indexes

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
aci	-	-	-	Yes	-	4000
cn	-	Yes	-	-	Yes	4000
dn2id	Non-configurable internal index					
ds-sync-conflict	-	Yes	-	-	-	4000
ds-sync-hist	-	-	Yes	-	-	4000
entryUUID	-	Yes	-	-	-	4000
givenName	-	Yes	-	-	Yes	4000
id2children	Non-configurable internal index					
id2subtree	Non-configurable internal index					
mail	-	Yes	-	-	Yes	4000
member	-	Yes	-	-	-	4000
objectClass	-	Yes	-	-	-	4000
sn	-	Yes	-	-	Yes	4000

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
telephone-Number	-	Yes	-	-	Yes	4000
uid	-	Yes	-	-	-	4000
uniqueMember	-	Yes	-	-	-	4000

When you create a new backend using the **dsconfig** command, OpenDJ directory servers create the following indexes automatically:

aci presence
ds-sync-conflict equality
ds-sync-hist ordering
entryUUID equality
objectClass equality

You can create additional indexes as described in "Configuring and Rebuilding Indexes".

Chapter 8
Managing Data Replication

OpenDJ software uses advanced data replication with automated conflict resolution to help ensure your directory services remain available during administrative operations that take an individual server offline, or in the event a server crashes or a network goes down. This chapter explains how to manage OpenDJ directory data replication. In this chapter you will learn to:

- Understand how replication operates in order to configure it appropriately

- Configure, initialize, and stop data replication

- Configure standalone directory servers and replication servers, or break a server that plays both roles into two standalone servers

- Configure replication groups, read-only replicas, assured replication, subtree replication, and fractional replication for complex deployments

- Configure and use change notification to synchronize external applications with changes to directory data

- Recover from situations where a user error has been applied to all replicas

- Resolve replication conflicts

8.1. About Replication

Read this section to learn more about how OpenDJ replication works.

8.1.1. Replication Defined

Replication is the process of copying updates between OpenDJ directory servers such that all servers converge on identical copies of directory data. Replication is designed to let convergence happen over time by default. (Assured replication can require, however, that the convergence happen before the client application is notified that the operation was successful.) Letting convergence happen over time means that different replicas can be momentarily out of sync. It also means that if you lose an individual server or even an entire data center, your directory service can keep on running, and then get back in sync when the servers are restarted or the network is repaired.

Replication is specific to OpenDJ software. Replication uses a specific protocol that replays update operations quickly, storing enough historical information about updates to resolve most conflicts

automatically. For example, if two client applications separately update a user entry to change the phone number, replication can identify the latest change, and apply it on all replicas without human intervention. To prevent it from growing forever, OpenDJ servers purge historical information periodically (default interval: three days). As a directory administrator, you must make sure that the servers do not purge historical information more often than you back up the directory data.

Keep server clocks synchronized for your topology. You can use NTP, for example. Keeping server clocks synchronized helps prevent issues with secure connections and with replication itself. Keeping server clocks synchronized also makes it easier to compare timestamps from multiple servers.

8.1.2. Replication Per Suffix

The primary unit of replication is the suffix, specified by a base DN such as `dc=example,dc=com`. (When you configure partial and fractional replication, however, you can replicate only part of a suffix, or only certain attributes on entries. Also, if you split your suffix across multiple backends, then you need to set up replication separately for each part of suffix in a different backend.) Replication also depends on the directory schema, defined on `cn=schema`, and the `cn=admin data` suffix with administrative identities and certificates for protecting communications. That content gets replicated as well.

The set of OpenDJ servers replicating data for a given suffix is called a *replication topology*. You can have more than one replication topology. For example, one topology could be devoted to `dc=example,dc=com`, and another to `dc=example,dc=org`. OpenDJ servers serve more than one suffix, and participate in more than one replication topology as shown in "Correct Replication Topology Configuration".

Correct Replication Topology Configuration
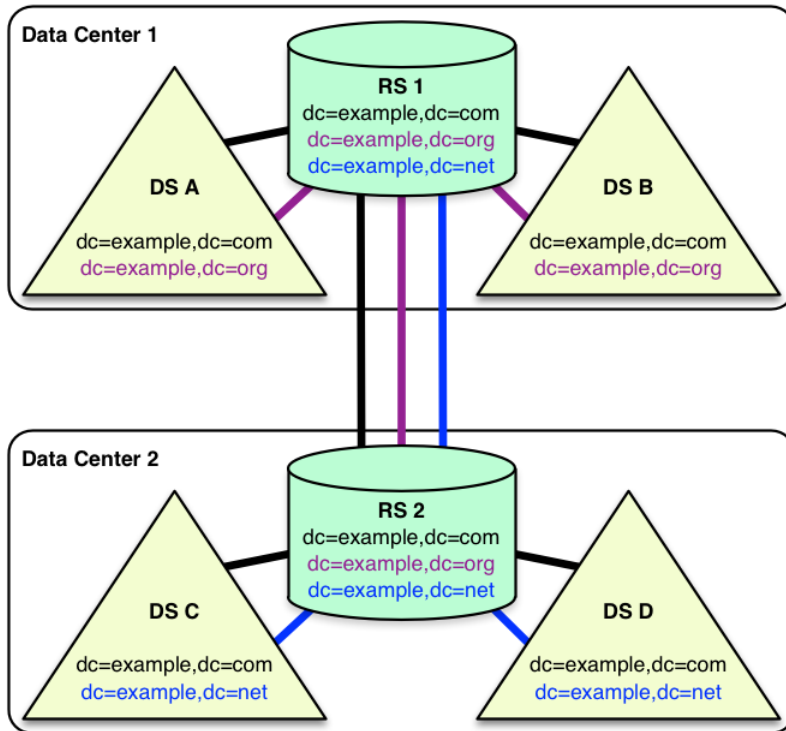
Within a replication topology, the suffixes being replicated are identified to the replication servers by their DNs. All the replication servers are fully connected in a topology. Consequently it is impossible to have multiple separate, independent topologies for data under the same DN within the overall set of servers. This is illustrated in "Incorrect Replication Topology Configuration".

Incorrect Replication Topology Configuration
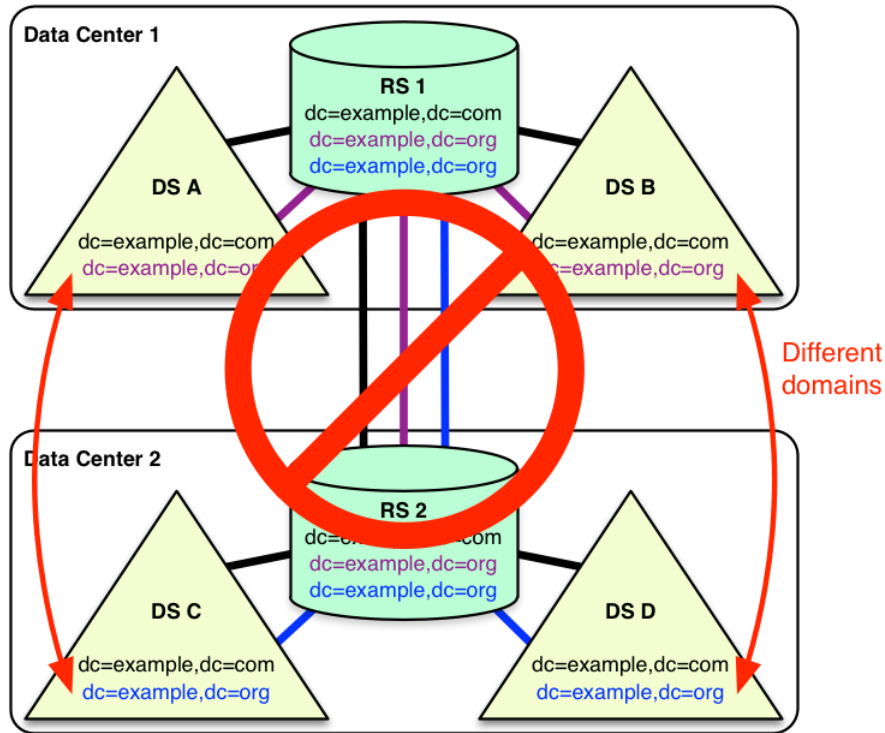
8.1.3. Replication Connection Selection

In order to understand what happens when individual servers stop responding due to a network partition or a crash, know that OpenDJ servers can simultaneously provide directory services and replication services. The two services are not the same, even if they can run alongside each other in the same OpenDJ server in the same JVM.

Replication relies on the replication service provided by OpenDJ replication servers, where OpenDJ directory servers publish changes made to their data, and subscribe to changes published by other OpenDJ directory servers. The replication service manages replication data only, handling replication traffic with OpenDJ directory servers and with other replication servers, receiving, sending, and storing only changes to directory data rather than directory data itself. Once a replication server is connected to a replication topology, it maintains connections to all other replication servers in that topology.

The directory service handles directory data. It responds to requests, and stores directory data and historical information. For each replicated suffix, such as `dc=example,dc=com`, `cn=schema`, and `cn=admin data`, the directory service publishes changes to a replication service, and subscribes to changes from that replication service. (Directory services do not publish changes directly to other directory services.) A directory server also resolves any conflicts that arise when reconciling changes from other directory servers, using the historical information about changes to resolve the conflicts. (Conflict resolution is the responsibility of the directory server rather than the replication server.)

Once a directory server is connected to a replication topology for a particular suffix, it connects to one replication server at a time for that suffix. The replication server provides the directory server with a list of all replication servers for that suffix. Given the list of possible replication servers it can connect to, the directory server can determine which replication server to connect to when starting up, or when the current connection is lost or becomes unresponsive.

For each replicated suffix, a directory server prefers to connect to a replication server:

1. In the same group as the directory server

2. Having the same initial data for the suffix as the directory server

3. If initial data was the same, a replication server having all the latest changes from the directory server

4. Running in the same JVM as the directory server

5. Having the most available capacity relative to other eligible replication servers

   Available capacity depends on how many directory servers in the topology are already connected to a replication server, and what proportion of all directory servers in the topology ought to be connected to the replication server.

   To determine what proportion of the total number of directory servers should be connected to a replication server, an OpenDJ server uses replication server weight. When configuring a replication server, you can assign it a weight (default: 1). The weight property takes an integer that indicates capacity to provide replication service relative to other servers. For example, a weight of 2 would indicate a replication server that can handle twice as many connected servers as a replication server with weight 1.

   The proportion of directory servers in a topology that should be connected to a given replication server is equal to (replication server weight)/(sum of replication server weights). In other words, if there are four replication servers in a topology each with default weights, the proportion for each replication server is 1/4.

Consider a situation where seven directory servers are connected to replication servers A, B, C, and D for `dc=example,dc=com` data. Suppose two directory servers each are connected to A, B, and C, one directory server is connected to replication server D. Replication server D is therefore the server with the most available capacity relative to other replication servers in the topology. All other criteria being equal, replication server D is the server to connect to when an eighth directory server joins the topology.

The directory server regularly updates the list of replication servers in case it must reconnect. As available capacity of replication servers for each replication topology can change dynamically, a directory server can potentially reconnect to another replication server to balance the replication load in the topology. For this reason, the server can also end up connected to different replication servers for different suffixes.

8.2. Configuring Replication Settings

This section shows how to configure replication with command-line tools, such as the **dsreplication** command, described in dsreplication(1) in the *Reference*.

8.2.1. Configuring Replication

You can configure a server to replicate with other servers by using the **dsreplication configure** command:

```
$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
 --host2 replica.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --trustAll \
 --no-prompt
...
Replication has been successfully configured.
...
```

> **Important**
>
> When you set **dsreplication** hostname options, such as `--hostname`, `--host1`, and `--host2`, make sure to always use either FQDNs or IP addresses. Use whichever are most stable and well-known to servers in your environment. (Never use local-only names or addresses, such as `localhost`, for production systems as they do not uniquely identify the server to other servers.)
>
> Make sure that you use the same hostname value that was provided at installation time to the **setup** command. If you do not know the original hostname used at installation time, read the entry with DN `cn=DIGEST-MD5,cn=SASL Mechanisms,cn=config`. Its `ds-cfg-server-fqdn` attribute holds the original hostname.
>
> Furthermore, always use the same hostname value for the same server each time you invoke the **dsreplication** command. The server uses the values you specify in its configuration. When you use one value to configure

replication and another to unconfigure replication, the latter operation fails on remote servers, leaving the system in a broken state.

If you do not know the original hostname values used with the **dsreplication** command, read the entry with DN `cn=all-servers,cn=Server Groups,cn=admin data`. Its `uniqueMember` attribute holds the hostname values.

To enable secure connections for replication use the `--secureReplication1`, and `--secureReplication2` options.

At this point, replication is configured but not active. You must initialize replication in order to replicate data with other servers.

> **Tip**
>
> When scripting the configuration to set up multiple replicas in quick succession, use the same initial replication server each time you run the command. In other words, pass the same `--host1`, `--port1`, `--bindDN1`, `--bindPassword1`, and `--replicationPort1` options for each of the other replicas that you set up in your script.

To replicate with another server, use **dsreplication enable** with the new server as the second server.

To Configure Replication Interactively

You can use the **dsreplication** command interactively.

> **Tip**
>
> When setting up trust between replicas with unrecognized certificates, such as the default self-signed certificates used for replication, the order makes a difference. You must *specify the remote server as the "first server"* or else the command will fail with the following message:
>
> ```
> There is an error with the certificate presented by the server.
> ```
>
> This is the case even when you trust the unrecognized certificate interactively "for this session only."

• Run the **dsreplication** command interactively by starting it without arguments.

  Adapt the following example as appropriate for your deployment:

```
$ dsreplication
What do you want to do?

    1)  Configure Replication
    2)  Unconfigure Replication
    3)  Initialize Replication on one Server
    4)  Initialize All Servers
    5)  Pre External Initialization
    6)  Post External Initialization
    7)  Display Replication Status
    8)  Purge Historical
    9)  Re-synchronizes the change-log changenumber on one server with the
        change-log changenumber of another.
```

```
    c)  cancel

Enter choice: 1


>>>> Specify server administration connection parameters for the first server

Directory server hostname or IP address [local.example.com]: remote.example.com

Directory server administration port number [4444]:

How do you want to trust the server certificate?

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

Enter choice [3]:

Global Administrator User ID, or bind DN if no Global Administrator is defined
[admin]: cn=Directory Manager

Password for user 'cn=Directory Manager':

Server Certificate:


User DN  : CN=remote.example.com, O=Administration Connector RSA
Self-Signed Certificate
Validity : From '<start-date>'
           To '<end-date>'
Issuer   : CN=remote.example.com, O=Administration Connector RSA
Self-Signed Certificate


Do you trust this server certificate?

    1)  No
    2)  Yes, for this session only
    3)  Yes, also add it to a truststore
    4)  View certificate details

Enter choice [2]:
Replication port for the first server (the port must be free) [8989]:

Do you want replication to use encrypted communication when connecting to
replication port 8989 on the first server? (yes / no) [no]: yes



>>>> Specify server administration connection parameters for the second server

Directory server hostname or IP address [local.example.com]: local.example.com

Directory server administration port number [4444]:

How do you want to trust the server certificate?
```

```
    1)  Automatically trust
    2)  Use a truststore
    3)  Manually validate

Enter choice [3]:

Global Administrator User ID, or bind DN if no Global Administrator is defined
[admin]: cn=Directory Manager

Password for user 'cn=Directory Manager':
Replication port for the second server (the port must be free) [8989]:

Do you want replication to use encrypted communication when connecting to
replication port 8989 on the second server? (yes / no) [no]: yes


Global Administrator must be created.
You must provide the credentials of the Global Administrator that will be
created to manage the server instances that are being replicated.
Global Administrator User ID [admin]:

Global Administrator Password:

Confirm Password:


You must choose at least one Base DN to be replicated.
Replicate base DN dc=example,dc=com? (yes / no) [yes]:

...
Replication has been successfully configured.  Note that for replication to work
you must initialize the contents of the base DNs that are being replicated
(use dsreplication initialize to do so)
.
...
```

8.2.2. Initializing Replicas

You can initialize replication between servers by performing initialization over the network after you
have configured replication, or by importing the same LDIF data on all servers and then configuring
replication. You can also add a new server by restoring backup of an existing replica onto the new
server and then configuring replication with an existing replica.

The alternatives are described step-by-step in the following procedures:

• "To Initialize Replication Over the Network"

• "To Initialize All Servers From the Same LDIF"

• "To Create a New Replica From an Existing Backup"

• "To Restore All Replicas to a Known State"

To Initialize Replication Over the Network

Initialization over the network while the server is online works well when you have no initial data, or when your network bandwidth is large compared to the initial amount of data to replicate.

1. Configure replication on all servers.

   See "Configuring Replication" for instructions.

2. Start replication with the **dsreplication initialize-all** command:

```
$ dsreplication \
 initialize-all \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --hostname opendj.example.com \
 --port 4444 \
 --trustAll \
 --no-prompt
```

To Initialize All Servers From the Same LDIF

Use this procedure when you have the same large LDIF available on all directory servers' local disks:

1. Configure replication for all servers.

   > **Important**
   >
   > Configuring replication overwrites data on the destination replica with data from the source replica, including administrative data.
   >
   > If the destination server replica generated encryption keys before replication was enabled, the destination server's encryption keys are overwritten when the administrative data is substituted with administrative data from the source server. *Any data encrypted with the destination server's old keys can no longer be decrypted.*
   >
   > Once replication is configured, administrative data is also replicated. If you use data confidentiality to protect data stored on disk, then replication must be configured before you import data to allow the replicas to share rather than overwrite each others' encryption keys.

   See "Configuring Replication" for instructions.

2. (Optional) If you have not already done so, enable data confidentiality as described in "Encrypting Directory Data" and "To Encrypt External Change Log Data".

3. Import the same LDIF on all servers as described in "To Import LDIF Data".

Do not yet accept updates to the directory data. "Read-Only Replicas" shows how to prevent replicas from accepting updates from clients.

4. Allow updates to the directory data by setting `writability-mode:enabled` using a command like the one you found in "Read-Only Replicas".

To Create a New Replica From an Existing Backup

You can create a new replica from a backup of an existing replica:

1. Install a new server to use as the new replica.

2. Backup the database on an existing server as described in "Backing Up Directory Data".

   At this point, other servers in the topology can continue to process updates.

3. Configure replication for the new replica:

```
$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
 --host2 new-replica.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --trustAll \
 --no-prompt
```

   At this point, you do not need to use the **dsreplication initialize** command.

4. On the new server, restore the database from the backup archive as described in "To Restore a Replica".

   As long as you restore the database on the new replica before the replication purge delay runs out, updates processed by other servers after you created the backup are replicated to the new server after you restore the data.

To Restore All Replicas to a Known State

OpenDJ replication is designed to make directory data converge across all replicas in a topology. Directory replication mechanically applies new changes to ensure that replicated data is the same everywhere, with newer changes taking precedence over older changes.

When you restore older backup data, for example, directory replication applies newer changes to the older data. This behavior is a good thing when the newer changes are correct.

This behavior can be problematic in the following cases:

• A bug or serious user error results in unwanted new changes that are hard to fix.

• The data in a test or proof-of-concept environment must regularly be reinitialized to a known state.

The **dsreplication** command has the following subcommands that let you reinitialize directory data, preventing replication from replaying changes that occurred before reinitialization:

• The **dsreplication pre-external-initialization** command removes the setting for the *generation ID* across the topology for a specified base DN. The generation ID is an internal-use identifier that replication uses to determine what changes to apply. This halts replication.

• The **dsreplication post-external-initialization** command sets a new generation ID across the topology, effectively resuming replication.

> **Caution**
>
> The steps in this procedure reinitialize the replication changelog, eliminating the history of changes that occurred before replication resumed. The replication changelog is described in "Change Notification For Your Applications". Applications that depend on the changelog for change notifications must be reinitialized after this procedure is completed.

1. (Optional) Prevent changes to the affected data during the procedure, as such changes are lost for the purposes of replication.

   For example, make each replica read-only as described in "Read-Only Replicas".

2. On a single server in the topology, run the **dsreplication pre-external-initialization** command for the base DN holding the relevant data, as shown in the following example:

   ```
   $ dsreplication \
    pre-external-initialization \
    --adminUID admin \
    --adminPassword password \
    --hostname opendj.example.com \
    --port 4444 \
    --baseDN dc=example,dc=com \
    --trustAll \
    --no-prompt
   ```

   Replication halts as the command takes effect.

   Changes made at this time are not replicated, even after replication resumes.

3. On each server in the topology, restore the data in the topology to the known state in one of the following ways:

- Import the data from LDIF as described in "To Import LDIF Data".

- Restore the data from backup as described in "To Restore a Standalone Server".

4. On a single server in the topology, run the **dsreplication post-external-initialization** command for the base DN holding the relevant data, as shown in the following example:

```
$ dsreplication \
 post-external-initialization \
 --adminUID admin \
 --adminPassword password \
 --hostname opendj.example.com \
 --port 4444 \
 --baseDN dc=example,dc=com \
 --trustAll \
 --no-prompt
```

Replication resumes as the command takes effect.

5. (Optional) If you made replicas read-only, make them read-write again by setting `writability-mode:enabled`.

8.2.3. Stopping Replication

How you stop replication depends on whether the change is meant to be temporary or permanent.

To Stop Replication Temporarily For a Replica

If you must stop a server from replicating temporarily, you can do so by using the **dsreplication suspend** and **dsreplication resume** commands.

> **Warning**
>
> Do not allow modifications on the replica for which replication is temporarily stopped. No record of such changes is kept, and the changes cause replication to diverge.

1. Run the **dsreplication suspend** command:

```
$ dsreplication \
 suspend \
 --hostname replica.example.com \
 --port 4444 \
 --adminUid admin \
 --adminPassword password \
 --trustAll \
 --no-prompt
```

2. (Optional) Run the **dsreplication resume** command:

```
$ dsreplication \
 resume \
 --hostname replica.example.com \
 --port 4444 \
 --adminUid admin \
 --adminPassword password \
 --trustAll \
 --no-prompt
```

To Stop Replication Permanently For a Replica

If you need to stop a server from replicating permanently, for example, when retiring a server, use the **dsreplication unconfigure** command.

> **Important**
>
> When you set **dsreplication** hostname options, such as `--hostname`, `--host1`, and `--host2`, make sure to always use either FQDNs or IP addresses. Use whichever are most stable and well-known to servers in your environment. (Never use local-only names or addresses, such as `localhost`, for production systems as they do not uniquely identify the server to other servers.)
>
> Make sure that you use the same hostname value that was provided at installation time to the **setup** command. If you do not know the original hostname used at installation time, read the entry with DN `cn=DIGEST-MD5,cn=SASL Mechanisms,cn=config`. Its `ds-cfg-server-fqdn` attribute holds the original hostname.
>
> Furthermore, always use the same hostname value for the same server each time you invoke the **dsreplication** command. The server uses the values you specify in its configuration. When you use one value to configure replication and another to unconfigure replication, the latter operation fails on remote servers, leaving the system in a broken state.
>
> If you do not know the original hostname values used with the **dsreplication** command, read the entry with DN `cn=all-servers,cn=Server Groups,cn=admin data`. Its `uniqueMember` attribute holds the hostname values.

1. Stop replication using the **dsreplication unconfigure** command:

```
$ dsreplication \
 unconfigure \
 --hostname new-replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --adminPassword password \
 --unconfigureAll \
 --trustAll \
 --no-prompt
```

The **dsreplication unconfigure** command completely removes the replication configuration information from the server.

2. (Optional) To restart replication for the server, run the **dsreplication configure** and **dsreplication initialize** commands again.

8.2.4. Standalone Replication Servers

OpenDJ replication is designed to be easy to implement in environments with a few servers, and scalable in environments with many servers. You can configure the replication service on each OpenDJ directory server in your deployment, for example, to limit the number of servers you deploy.

In a large deployment, you can use standalone replication servers—servers that do nothing but relay replication messages—to configure (and troubleshoot) the replication service separately from the directory service. You need only a few standalone replication servers publishing changes to serve many directory servers subscribed to the changes. Furthermore, replication is designed so a directory server needs to connect only to the nearest replication server in order to to replicate with all servers in the topology. Only the standalone replication servers participate in fully meshed replication.

All replication servers in a topology are connected to all other replication servers. Directory servers are connected only to one replication server at a time, and their connections should be to replication servers on the same LAN. Therefore, the total number of replication connections, $Total_{conn}$, is expressed as follows:

$$Total_{conn} = (N_{RS} * (N_{RS}-1))/2 + N_{DS} \tag{8.1}$$

Here, N_{RS} is the number of replication servers, and N_{DS} is the number of standalone directory servers.

In other words, if you have only three servers, then $Total_{conn}$ is three with no standalone servers. However, if you have two data centers, and need 12 directory servers, then with no standalone directory servers $Total_{conn}$ is (12 * 11)/2 or 66. Yet, with four standalone replication servers, and 12 standalone directory servers, $Total_{conn}$ is (4 * 3)/2 + 12, or 18. Only four of those connections extend over the WAN. (By running four directory servers that also run replication servers and eight standalone directory servers, you reduce the number of replication connections to 14 for 12 replicas.)

This is shown in "Deployment For Multiple Data Centers".

Deployment For Multiple Data Centers
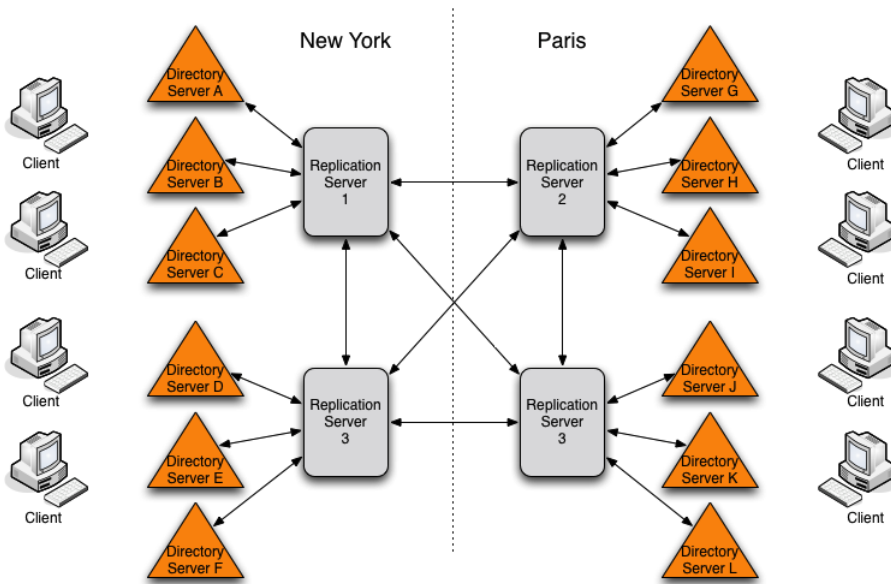
To Set Up a Standalone Replication Server

This example sets up a standalone replication server to handle the replication traffic between two directory servers that do not handle replication themselves.

Here the replication server is `rs-only.example.com`. The directory servers are `opendj.example.com` and `replica.example.com`.

In a real deployment, you would have more replication servers to avoid a single point of failure.

1. Set up the replication server as a directory server that has no database.

2. Set up the directory servers as standalone directory servers.

3. Configure replication with the `--noReplicationServer` or `--onlyReplicationServer` option:

```
$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
```

```
--bindDN1 "cn=Directory Manager" \
--bindPassword1 password \
--noReplicationServer1 \
--host2 rs-only.example.com \
--port2 4444 \
--bindDN2 "cn=Directory Manager" \
--bindPassword2 password \
--replicationPort2 8989 \
--onlyReplicationServer2 \
--trustAll \
--no-prompt

$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 replica.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --noReplicationServer1 \
 --host2 rs-only.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --onlyReplicationServer2 \
 --trustAll \
 --no-prompt
```

4. Initialize replication from one of the directory servers:

```
$ dsreplication \
 initialize-all \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --hostname opendj.example.com \
 --port 4444 \
 --trustAll \
 --no-prompt
```

8.2.5. Standalone Directory Server Replicas

When you configure replication for an OpenDJ directory server, you can give the directory server the capability to handle replication traffic as well. As described in "Standalone Replication Servers", OpenDJ servers can also be configured to handle only replication traffic.

Alternatively you can configure an OpenDJ directory server to connect to a remote replication server of either variety, but to remain only a directory server itself. This sort of standalone directory server replica is shown in "Deployment For Multiple Data Centers".

Furthermore, you can make this standalone directory server replica read-only for client applications, accepting only replication updates.

To Set Up a Standalone Directory Server Replica

The following steps show how to configure the server as a standalone, directory server-only replica of an existing replicated directory server.

1. Set up replication between other servers.

2. Install the directory server without configuring replication, but creating at least the base entry to be replicated.

3. Configure replication with the appropriate `--noReplicationServer` option:

```
$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --host2 ds-only.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --noReplicationServer2 \
 --trustAll \
 --no-prompt
```

Here the existing server is both directory server and replication server. If the existing server is a standalone replication server, then also use the appropriate `--onlyReplicationServer` option.

4. Initialize data on the new directory server replica:

```
$ dsreplication \
 initialize \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --hostSource opendj.example.com \
 --portSource 4444 \
 --hostDestination ds-only.example.com \
 --portDestination 4444 \
 --trustAll \
 --no-prompt
```

5. If you want to make the directory server replica read-only for client application traffic, see "Read-Only Replicas".

8.2.6. Replication Groups

Replication lets you define groups so that replicas communicate first with replication servers in the group before going to replication servers outside the group. Groups are identified with unique numeric group IDs.

Replication groups are designed for deployments across multiple data centers, where you aim to focus replication traffic on the LAN rather than the WAN. In multi-data center deployments, group nearby servers together.

To Set Up Replication Groups

For each group, set the appropriate group ID for the topology on both the replication servers and the directory servers.

The example commands in this procedure set up two replication groups, each with a replication server and a directory server. The directory servers are `opendj.example.com` and `replica.example.com`. The replication servers are `rs.example.com` and `rs2.example.com`. In a full-scale deployment, you would have multiple servers of each type in each group. All directory servers and replication servers in a data center would belong to the same group.

1. Pick a group ID for each group.

   The default group ID is 1.

2. Set the group ID for each group by replication domain on the directory servers:

   ```
   $ dsconfig \
    set-replication-domain-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --provider-name "Multimaster Synchronization" \
    --domain-name "dc=example,dc=com" \
    --set group-id:1 \
    --trustAll \
    --no-prompt

   $ dsconfig \
    set-replication-domain-prop \
    --hostname replica.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --provider-name "Multimaster Synchronization" \
    --domain-name "dc=example,dc=com" \
    --set group-id:2 \
    --trustAll \
    --no-prompt
   ```

3. Set the group ID for each group on the replication servers:

```
$ dsconfig \
 set-replication-server-prop \
 --hostname rs.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set group-id:1 \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-replication-server-prop \
 --hostname rs2.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set group-id:2 \
 --trustAll \
 --no-prompt
```

8.2.7. Read-Only Replicas

By default all directory servers in a replication topology are read-write. You can, however, choose to make replicas take updates only from the replication protocol, and refuse updates from client applications:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set writability-mode:internal-only \
 --trustAll \
 --no-prompt
```

8.2.8. Assured Replication

In standard replication, when a client requests an update operation the directory server performs the update, and, if the update is successful, sends information about the update to the replication service, and sends a result code to the client application right away. As a result, the client application can conclude that the update was successful, *but only on the replica that handled the update*.

Assured replication lets you force the replica performing the initial update to wait for confirmation that the update has been received elsewhere before sending a result code to the client application.

You can configure assured replication either to wait for one or more replication servers to acknowledge reception, or to wait for all directory servers to replay the update.

Assured replication is slower than standard replication, potentially waiting for a timeout before failing when the network or other servers are down.

When working with assured replication, the replication server property `degraded-status-threshold` (default: 5000), sets the number of operations allowed to build up in the replication queue before the server is assigned degraded status. When a replication server has degraded status, *assured replication ceases to have an effect.* Bear in mind that assured replication can break under heavy load.

To Ensure Updates Reach Replication Servers

Safe data mode requires the update be sent to `assured-sd-level` replication servers before acknowledgement is returned to the client application:

- For each directory server, set safe data mode for the replication domain, and set the safe data level:

```
$ dsconfig \
 set-replication-domain-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set assured-type:safe-data \
 --set assured-sd-level:1 \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-replication-domain-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set assured-type:safe-data \
 --set assured-sd-level:1 \
 --trustAll \
 --no-prompt
```

To Ensure Updates Are Replayed Everywhere

Safe read mode requires the update be replayed on all directory servers before acknowledgement is returned to the client application:

- For each directory server, set safe read mode for the replication domain:

```
$ dsconfig \
 set-replication-domain-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set assured-type:safe-read \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-replication-domain-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set assured-type:safe-read \
 --trustAll \
 --no-prompt
```

8.2.9. Subtree Replication

OpenDJ servers can perform subtree replication. For example, they can replicate `ou=People,dc=example,dc=com`, but not the rest of `dc=example,dc=com`, by putting the subtree in a separate backend from the rest of the suffix.

In this example, you might have a `userRoot` backend containing everything in `dc=example,dc=com` except `ou=People,dc=example,dc=com`, and a separate `peopleRoot` backend for `ou=People,dc=example,dc=com`. You can then configure OpenDJ servers to replicate `ou=People,dc=example,dc=com` in a separate topology.

For details, see "Splitting Data Across Multiple Backends".

8.2.10. Fractional Replication

OpenDJ servers can perform fractional replication. For fractional replication, you specify the attributes to include in the replication process, or the attributes to exclude from the replication process.

You set fractional replication configuration as `fractional-include` or `fractional-exclude` properties of a replication domain. When you include attributes, the attributes that are required on the relevant object classes are also included, whether you specify them or not. When you exclude attributes, the excluded attributes must be optional attributes for the relevant object classes. Fractional replicas still respect schema definitions.

Fractional replication filters objects at the replication server level. Each attribute must remain available on at least one replica in the topology. Fractional replication is not designed to exclude the same attribute on every replica in a topology. When you configure a replica to exclude an attribute, the directory server checks that the attribute is never added to the replica as part of any LDAP operation. As a result, if you exclude the attribute everywhere, it can never be added anywhere.

When using fractional replication, initialize replication as you would normally. You cannot create a full replica, however, from a replica with only a subset of the data. If you must prevent data from being replicated across a national boundary, for example, split the replication server that handles updates from the directory servers as described in "To Set Up a Standalone Replication Server".

For example, you might configure an externally facing fractional replica to include only some `inetOrgPerson` attributes:

```
$ dsconfig \
 set-replication-domain-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set fractional-include:inetorgperson:cn,givenname,mail,mobile,sn,telephonenumber \
 --trustAll \
 --no-prompt
```

As another example, you might exclude a custom attribute called `sessionToken` from being replicated:

```
$ dsconfig \
 set-replication-domain-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set fractional-exclude:*:sessionToken \
 --trustAll \
 --no-prompt
```

The latter example only has an effect if you first define a `sessionToken` attribute in the directory server schema.

8.2.11. Breaking a Multi-Role Server Into Standalone Components

As described in "About Replication", a replication topology is made up of servers playing the role of directory server, and servers playing the role of replication server. By default, each replicated OpenDJ server plays both roles. Some deployments call for standalone directory servers and

standalone replication servers, however. (In practice, "standalone" technically usually refers only to the role with respect to replication of user data. In fact standalone servers generally continue to play both roles for server configuration data under `cn=admin data` and `cn=schema`. The update traffic to these suffixes is, however, generally orders of magnitude lower than update traffic for user data.)

If possible avoid breaking apart an existing multi-role server. Instead, set up standalone servers as described in "Standalone Replication Servers" and "Standalone Directory Server Replicas".

The following procedure breaks a multi-role server into two standalone servers while preserving existing data. It does require disk space initially to hold copies of existing data.

To Break a Multi-Role Server Into Standalone Components

The following steps show how to break a multi-role OpenDJ server into a standalone directory server and a standalone replication server.

While you carry out this procedure, do not allow any client traffic to the servers you modify.

1. Make sure you have already set up at least a couple of OpenDJ servers that replicate user data.

    This example starts with the following multi-role servers:

    - `/path/to/dsrs1` (ports: 1389, 1636, 4444, 8989; replicating user data for `dc=example,dc=com`)

    - `/path/to/dsrs2` (ports: 2389, 2636, 5444, 9989; replicating user data for `dc=example,dc=com`)

    `/path/to/dsrs1` is the target server to be broken into standalone components.

    When you begin, the target server has both directory server and replication server components.

    Before you proceed:

    - Read the rest of the procedure, and make sure you understand the steps.

    - Direct client traffic away from the target server.

    - Back up the target server.

2. Run the **dsreplication status** command before making changes:

```
$ dsreplication \
 status \
 --port 4444 \
 --hostname opendj.example.com \
 --adminUID admin \
 --adminPassword password \
 --baseDN "cn=admin data" \
 --baseDN cn=schema \
 --baseDN dc=example,dc=com \
 --trustAll \
 --no-prompt

Suffix DN          :...: DS ID : RS ID :..
.
------------------:...:-------:-------:...
cn=admin data      :...: 29388 : 32560 :...
cn=admin data      :...: 7044  : 29137 :...
cn=schema          :...: 24612 : 32560 :...
cn=schema          :...: 22295 : 29137 :...
dc=example,dc=com :...: 20360 : 32560 :...
dc=example,dc=com :...: 12164 : 29137 :..
.
...
```

Keep the output of the command for the IDs shown. The information is used later in this procedure.

3. Temporarily disable the multimaster synchronization provider on the target server:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

This step is also shown in "To Stop Replication Temporarily For a Replica".

4. Temporarily disable the backend holding the replicated data:

```
$ dsconfig \
 set-backend-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

5. Stop the target server:

```
$ stop-ds
Stopping Server..
.
... msg=The Directory Server is now stopped
```

6. Make two copies of the server files:

```
$ cd /path/to/
```

One copy will be the standalone directory server:

```
$ cp -r dsrs1 ds
```

The other copy will the standalone replication server:

```
$ cp -r dsrs1 rs
```

7. Start the copy that will become the standalone directory server, remove the replication server and changelog configuration, enable the user data backend, and then enable the multimaster synchronization provider on the directory server:

```
# The following command removes the replication server configuration.

dsconfig \
 delete-replication-server \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --trustAll \
 --no-prompt
```

```
# The following command disables the changelog for the user data
# in dc=example,dc=com.

dsconfig \
 set-external-changelog-domain-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com
 --set enabled:false
 --trustAll \
 --no-prompt

# The following command enables the user data backend.

dsconfig \
 set-backend-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set enabled:true \
 --trustAll \
 --no-prompt

# The following command enables the multimaster synchronization provider.

dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

You can then remove the files for the changelog on the directory server:

```
$ rm /path/to/ds/changelogDb/*
```

8. If the replication server is on the same host as the directory server, carefully change the connection handler port numbers and the administration port number in the configuration file before starting the replication server. Before making any changes, make sure that the new port numbers you use are available, and not in use by any other services on the system.

Change the port numbers for the LDAP and LDAPS connection handlers as described in "To Change the LDAP Port Number".

The following example changes the administration port to 6444. After this command succeeds, you must restart the server in order to use the **dsconfig** command again:

```
$ dsconfig \
 set-administration-connector-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set listen-port:6444 \
 --trustAll \
 --no-prompt
```

Restart the server to be able to connect on the new administration port:

```
$ stop-ds --restart
Stopping Server..
.
..
.
...The Directory Server has started successfully
```

9. Change the server ID values for the `cn=admin data` and `cn=schema` replication domains on the copy that is to become the standalone replication server.

Replication uses unique server IDs to distinguish between different directory server replicas. When you make identical copies of the original multi-role server, the server IDs on the new standalone directory server and on the new standalone replication server are identical.

For the user data replication domains, such as `dc=example,dc=com`, you are going to fix the duplicate server ID problem as part of this procedure. When you remove the replication domain configuration information from the new standalone replication server for user data, part of the configuration information that you remove is the server ID. For the administrative data and directory schema, however, the new standalone replication server must maintain its administrative and schema data in sync with other servers, so it still holds that data like any other directory server. The server IDs for the `cn=admin data` and `cn=schema` replication domains must therefore be changed so as not to conflict with other existing server IDs.

If you try to edit server IDs by using the **dsconfig** command, you encounter an error:

```
The Replication Domain property "server-id" is read-only and cannot be
modified
```

You must instead edit the server ID values directly in the configuration file while the new standalone replication server is stopped.

Before editing the configuration file, refer to the information you gather in Step 2 for the list of IDs that are in use in the replication topology. You must choose server ID values that are unique, and that are between 0 and 65535 inclusive.

After choosing two valid, unused server ID values, carefully edit the configuration file, `/path/to/rs/config/config.ldif`, to change the `ds-cfg-server-id` values for the entries with DNs `cn=cn=admin data,cn=domains,cn=Multimaster Synchronization,cn=Synchronization Providers,cn=config` and `cn=cn=schema,cn=domains,cn=Multimaster Synchronization,cn=Synchronization Providers,cn=config`.

For example, if the duplicate server IDs were 29388 and 24612, and you edited the configuration file to use 12345 and 23456 instead, the result might appear as follows:

```
$ grep -B 1 ds-cfg-server-id /path/to/rs/config/config.ldif
cn: cn=admin data
#ds-cfg-server-id: 29388
ds-cfg-server-id:
 12345
--
cn: cn=schema
#ds-cfg-server-id: 24612
ds-cfg-server-id: 23456
```

10. Start the copy that is to become the standalone replication server, remove the user data backend configuration, remove the replication domain for the user data, and then enable the multimaster synchronization provider on the directory server:

```
# The following command removes the user data backend configuration.

dsconfig \
 delete-backend \
 --port 6444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --trustAll \
 --no-prompt

# The following command removes the replication domain for the user data.

dsconfig \
 delete-replication-domain \
 --port 6444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --trustAll \
 --no-prompt

# The following command enables the multimaster synchronization provider.
```

```
dsconfig \
 set-synchronization-provider-prop \
 --port 6444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

You can then remove the files for the user data backend on the replication server:

```
$ rm -rf /path/to/rs/db/userRoot
```

11. If you have moved servers with secure ports configured, the host names in the server certificates might no longer correspond to the new host names.

    For details see "*Changing Server Certificates*".

12. After testing that everything is working to your satisfaction, you can allow normal client traffic to the new directory server, and retire the old multi-role server (**rm -rf /path/to/dsrs1** in this example).

8.3. Change Notification For Your Applications

Some applications require notification when directory data updates occur. For example, an application might need to sync directory data with another database, or the application might need to kick off other processing when certain updates occur.

In addition to supporting persistent search operations, OpenDJ provides an external change log mechanism to allow applications to be notified of changes to directory data.

This section includes the following procedures:

- "To Enable the External Change Log"

- "To Encrypt External Change Log Data"

- "To Use the External Change Log"

- "To Allow a User to Read the Change Log"

- "To Include Unchanged Attributes in the External Change Log"

- "To Limit External Change Log Content"

- "To Align Draft Change Numbers"

• "To Enable the External Change Log (Standalone Server)"

To Enable the External Change Log

OpenDJ directory servers that are not replicated do not publish an external change log. A directory server that publishes a change log must function as both a directory server, and a replication server for the suffix whose changes you want logged.

• Configure replication without using the `--noReplicationServer` or `--onlyReplicationServer` option.

With replication configured, the data is under `cn=changelog`. The user reading the changelog must have appropriate access, and must have the `changelog-read` privilege. Directory Manager is not subject to access control, and has the privilege. The following example shows that Directory Manager can read the changelog:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN cn=changelog \
 --searchScope base \
 "(&)"
dn: cn=changelog
objectClass: top
objectClass: container
cn: changelog
```

If the user reading the changelog is not Directory Manager, see "To Allow a User to Read the Change Log".

To Encrypt External Change Log Data

OpenDJ servers do not encrypt external change log data by default. This means that any user with system access to read directory files can potentially access external change log data in cleartext.

In addition to preventing read access by other users as described in "Setting Up a System Account for an OpenDJ Server", you can configure confidentiality for external change log data. When confidentiality is enabled, the server encrypts change log records before storing them.

> **Important**
>
> Encrypting stored directory data does not prevent it from being sent over the network in the clear.
>
> Apply the suggestions in "*Securing Network Connections*" in the *Security Guide* to protect data sent over the network.

OpenDJ servers encrypt data using a symmetric key that is stored with the server configuration. The symmetric key is encrypted in turn with the server's public key that is also stored with the server

configuration. When multiple servers are configured to replicate data as described in "Configuring Replication Settings", the servers replicate the keys as well, allowing any server replica to decrypt the data.

Encrypting and decrypting data comes with costs in terms of cryptographic processing that reduces throughput and of extra space for larger encrypted values. In general, tests with default settings show that the cost of enabling confidentiality can be quite modest, but your results can vary based on your systems and on the settings used for `cipher-transformation` and `cipher-key-length`. Make sure you test your deployment to qualify the impact of confidentiality before enabling it in production.

Follow this procedure to enable confidentiality:

1. Before you enable confidentiality on a replication server for the external change log data, first enable confidentiality for data stored in directory backends.

   For details, see "Encrypting Directory Data".

2. Enable backend confidentiality with the default encryption settings as shown in the following example:

```
$ dsconfig \
 set-replication-server-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set confidentiality-enabled:true \
 --no-prompt \
 --trustAll
```

   Encryption applies to the entire change log regardless of the confidentiality settings for each domain.

   After confidentiality is enabled, new change log records are encrypted. OpenDJ servers do not rewrite old records in encrypted form.

3. (Optional) If necessary, adjust additional confidentiality settings.

   Use the same cipher suite for external change log confidentiality as was used to configure data confidentiality.

   The default settings for confidentiality are `cipher-transformation: AES/CBC/PKCS5Padding` and `cipher-key-length: 128`. This means the algorithm is the Advanced Encryption Standard (AES), the cipher mode is Cipher Block Chaining (CBC), and the padding is PKCS#5 padding as described in RFC 2898: PKCS #5: Password-Based Cryptography Specification. The syntax for the `cipher-transformation` is *algorithm/mode/padding*, and all three must be specified. When the algorithm does not require a mode, use `NONE`. When the algorithm does not require padding, use `NoPadding`. Use of larger `cipher-key-length` values can require that you install JCE policy files such as those for

unlimited strength, as described in "Using Unlimited Strength Cryptography" in the *Security Guide*.

To Use the External Change Log

You read the external change log over LDAP. In addition, when you poll the change log periodically, you can get the list of updates that happened since your last request.

The external change log mechanism uses an LDAP control with OID `1.3.6.1.4.1.26027.1.5.4` to allow the exchange of cookies for the client application to bookmark the last changes seen, and then start reading the next set of changes from where it left off on the previous request.

This procedure shows the client reading the change log as `cn=Directory Manager`. Make sure your client application reads the changes with sufficient access and privileges to view all the changes it needs to see.

1. Send an initial search request using the LDAP control with no cookie value:

    In this example, two changes appear in the changelog:

    ```
    $ ldapsearch \
     --port 1389 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password \
     --baseDN cn=changelog \
     --control "1.3.6.1.4.1.26027.1.5.4:false" \
     "(&)" \
     changes changeLogCookie targetDN
    # Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4): <COOKIE1>
    dn: replicationCSN=<CSN1>,dc=example,dc=com,cn=changelog
    changes::
     cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldyBkZXNjcmlwdGlvbgotCnJlcGxhY2U6IG1vZGlmaWVyc05hbWUKbW9kaWZpZpZX
    targetDN: uid=bjensen,ou=People,dc=example,dc=com
    changeLogCookie: <COOKIE1>

    # Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4): <COOKIE2>
    dn: replicationCSN=<CSN2>,dc=example,dc=com,cn=changelog
    changes::
     cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldywgaW1wcm92ZWQgZGVzY3JpcHRpb24KLQpyZXBsYWNlOiBtb2RpZmllcnNOYYW
    targetDN: uid=bjensen,ou=People,dc=example,dc=com
    changeLogCookie: <COOKIE2>
    ```

    The changes are base64-encoded. You can decode them using the **base64** command. The following example decodes the first change:

```
$ base64 decode --encodedData
 cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldyBkZXNjcmlwdGlvbgotCnJlcGxhY2U6IG1vZGlmaWVyc05hbWUKbW9kaWZpZXpX
replace: description
description: New description
-
replace: modifiersName
modifiersName: uid=bjensen,ou=People,dc=example
,dc=com
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>
-
```

Notice the `changeLogCookie` value, which has the form *base-dn:CSN*.

2. To start reading a particular change in the changelog, provide the cookie with the control:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "1.3.6.1.4.1.26027.1.5.4:false:$COOKIE1" \
 "(&)" \
 changes changeLogCookie targetDN
# Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4): <COOKIE2>
dn: replicationCSN=<CSN2>,dc=example,dc=com,cn=changelog
changes::
 cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldywgaW1wcm92ZWQgZGVzY3JpcHRpb24KLQpyZXBsYWNlOiBtb2RpZmllcnNOYYW
targetDN: uid=bjensen,ou=People,dc=example,dc=com
changeLogCookie: <COOKIE2>
```

The following command decodes the changes:

```
$ base64 decode --encodedData
 cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldywgaW1wcm92ZWQgZGVzY3JpcHRpb24KLQpyZXBsYWNlOiBtb2RpZmllcnNOYYW
replace: description
description: New, improved
 description
-
replace: modifiersName
modifiersName: uid=bjensen,ou=People,dc=example
,dc=com
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>
-
```

3. If you lose the cookie, start over from the earliest available change by sending a search request with no value for the cookie.

To Allow a User to Read the Change Log

For a user to read the changelog, the user must have access to read, search, and compare changelog attributes, might have access to use the control to read the external changelog, and must have the `changelog-read` privilege.

1. Give the user access to read and search the changelog.

   The following example adds a global ACI to give `My App` access to the changelog:

   ```
   $ dsconfig \
    set-access-control-handler-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --add global-aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*||+\")\
   (version 3.0; acl \"My App can access cn=changelog\"; \
   allow (read,search,compare) \
   userdn=\"ldap:///cn=My App,ou=Apps,dc=example,dc=com\";)" \
    --trustAll \
    --no-prompt
   ```

2. Give the user access to use the control.

   The following example adds a global ACI to give `My App` access to use the control:

   ```
   $ dsconfig \
    set-access-control-handler-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --add global-aci:"(targetcontrol=\"1.3.6.1.4.1.26027.1.5.4\")\
   (version 3.0; acl \"My App control access\"; \
   allow (read) \
   userdn=\"ldap:///cn=My App,ou=Apps,dc=example,dc=com\";)" \
    --trustAll \
    --no-prompt
   ```

3. Give the user the `changelog-read` privilege:

```
$ cat changelog-read.ldif
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: changelog-read

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 changelog-read.ldif
```

4. Check that the user can read the changelog:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "1.3.6.1.4.1.26027.1.5.4:false" \
 "(&)" \
 changes changeLogCookie targetDN
```

To Include Unchanged Attributes in the External Change Log

As shown above, the changes returned from a search on the external change log include only what was actually changed. If you have applications that need additional attributes published with every change log entry, regardless of whether or not the attribute itself has changed, then specify those using `ecl-include` and `ecl-include-for-deletes`.

1. Set the attributes to include for all update operations with `ecl-include`:

```
$ dsconfig \
 set-external-changelog-domain-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --set ecl-include:"@person" \
 --trustAll \
 --no-prompt
```

2. Set the attributes to include for deletes with `ecl-include-for-deletes`:

```
$ dsconfig \
 set-external-changelog-domain-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --add ecl-include-for-deletes:"*" \
 --add ecl-include-for-deletes:"+" \
 --trustAll \
 --no-prompt
```

To Limit External Change Log Content

Limit external change log content by disabling the domain for a base DN. By default, `cn=schema` and `cn=admin data` are not enabled.

• Prevent the server from logging changes by disabling the domain:

```
$ dsconfig \
 set-external-changelog-domain-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

To Align Draft Change Numbers

The external change log can be used by applications that follow the *Internet-Draft: Definition of an Object Class to Hold LDAP Change Records*, and that cannot use change log cookies shared across the replication topology. Nothing special is required to get the objects specified for this legacy format, but there are steps you must perform to align change numbers across replicas.

Change numbers described in the Internet-Draft are simple numbers, not cookies. When change log numbers are aligned across replicas, applications fail over from one replica to another when necessary.

If you do not align the change numbers, each server keeps its own count. The same change numbers can refer to different changes on different replicas.

For example, if you install a new replica and initialize replication from an existing server, the last change numbers are likely to differ. The following example shows different last change numbers for an existing server and for a new replica that has just been initialized from the existing replica:

```
$ ldapsearch \
 --hostname existing.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "" \
 --searchScope base \
 "(&)" lastChangeNumber
dn:
lastChangeNumber: 285924


Result Code:  0 (Success)

$ ldapsearch \
 --hostname new.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "" \
 --searchScope base \
 "(&)" lastChangeNumber
dn:
lastChangeNumber: 198643


Result Code:  0 (Success)
```

When you add a new replica to an existing topology, follow these steps to align the change numbers with those of an existing server.

These steps can also be used at any time to align the change numbers:

1. Make sure that the new replica has the same replication configuration as the existing replica.

    Specifically, both replicas must replicate the same suffixes in order for the change number calculations to be the same on both replicas. If the suffix configurations differ, the change numbers cannot be aligned.

2. (Optional) If you must start the new replica's change numbering from a specific change, determine the `changeNumber` to use.

    The `changeNumber` must be from a change that has not yet been purged according to the replication purge delay, which by default is three days.

3. Using the **dsreplication** command installed with the new replica, reset the change number on the new replica to the change number from the existing replica.

    The following example does not specify the change number to use. By default, the new replica uses the last change number from the existing replica:

```
$ dsreplication \
 reset-change-number \
 --adminUID admin \
 --adminPassword password \
 --hostSource existing.example.com \
 --portSource 4444 \
 --hostDestination new.example.com \
 --portDestination 4444 \
 --trustAll \
 --no-prompt
Change-log change number reset task has finished successfully.
...
```

At this point, the new replica's change log starts with the last change number from the existing replica. Earlier change numbers are no longer present in the new replica's change log.

To Enable the External Change Log (Standalone Server)

In development and test environments, you can enable the external change log on a single server, without actually replicating directory data. If you use replication and replicated servers, follow the instructions in "To Enable the External Change Log" instead.

Follow these steps to enable the external change log on a single standalone directory server:

1. Create a replication server configuration entry as in the following example:

```
$ dsconfig \
 create-replication-server \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set replication-port:8989 \
 --set replication-server-id:2 \
 --trustAll \
 --no-prompt
```

2. Create a replication domain configuration entry as in the following example:

```
$ dsconfig \
 create-replication-domain \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "Example.com" \
 --set base-dn:dc=example,dc=com \
 --set replication-server:opendj.example.com:8989 \
 --set server-id:3 \
 --trustAll \
 --no-prompt
```

At this point, you can begin using the external change log under `cn=changelog`.

8.4. Recovering From User Error

This section covers how to restore accidentally deleted or changed data.

Changes to a replicated OpenDJ directory service are similar to those made with the Unix **rm** command, but with a twist. With the **rm** command, if you make a mistake you can restore your files from backup, and lose only the work done since the last backup. If you make a mistake with a update to the directory service however, then after you restore a server from backup, replication efficiently replays your mistake to the server you restored.

There is more than one way to recover from user error. None of the ways involve simply changing OpenDJ settings. All of the ways instead involve manually fixing mistakes.

Consider these alternatives:

• Encourage client applications to provide end users with undo capability if necessary. In this case, client applications take responsibility for keeping an undo history.

• Maintain a record of each update to the service, so that you can manually "undo" mistakes.

  You can use the external change log. A primary advantage to the external change log is that the change log is enabled with replication, and so it does not use additional space.

  See "Change Notification For Your Applications" for instructions on enabling, using, and configuring the external change log. In particular, see "To Include Unchanged Attributes in the External Change Log" for instructions on saving not only what is changed, but also all attributes when an entry is deleted.

  OpenDJ servers can also write to a file-based audit log, but the audit log does not help with a general solution in this case. The OpenDJ audit log records changes to the data. When you delete an entry however, the audit log does not record the entry before deletion. The following example shows the audit log records of some changes made to Barbara Jensen's entry:

```
# <datestamp>; conn=<number>; op=<number>
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: This is the description I want.
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>

# <datestamp>; conn=<number>; op=<number>
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: I never should have changed this!
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>

# <datestamp>; conn=<number>; op=<number>
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: delete
```

You can use these records to fix the mistaken update to the description, but the audit log lacks the information needed to restore Barbara Jensen's deleted entry.

• For administrative errors that involve directory data, if you have properly configured the external change log, then use it.

If not, an alternative technique consists of restoring backup to a separate server not connected to the replication topology. (Do not connect the server to the topology as replication replays mistakes, too.) Compare data on the separate restored server to the live servers in the topology, and then fix the mistakes manually.

A more drastic alternative consists of rebuilding the entire service from backup, by unconfiguring replication and restoring all servers from backup (or restoring one server and initializing all servers from that one). This alternative is only recommended in the case of a major error where you have a very fresh backup (taken immediately before the error), and no client applications are affected.

• For administrative configuration errors that prevent servers from starting, know that OpenDJ servers keep a copy of the last configuration that allowed the server to start correctly. The file is `/path/to/opendj/config/config.ldif.startok`.

OpenDJ servers also back up earlier versions of the configuration under `/path/to/opendj/config/archived-configs/`.

You can therefore compare the current configuration with the earlier configurations, and repair mistakes manually (avoiding trailing white space at the end of LDIF lines) while the server is down.

8.5. Resolving Replication Conflicts

Replication is eventually consistent by design. Rather than dealing with some sort of distributed transaction mechanism that would make write operations impossible during a network partition, replication supports basic write availability. Changes are applied locally and then replayed to remote replicas. This means it is possible to have conflicts, however. A *replication conflict* arises when incompatible changes are applied concurrently to multiple read-write replicas.

Two types of conflicts can occur: *modify conflicts* and *naming conflicts*. Modify conflicts involve concurrent modifications to the same entry. Naming conflicts involve other operations that affect the DN of the entry.

Replication can resolve modify conflicts and many naming conflicts automatically by replaying the changes in the correct order. To determine the relative order in which changes occurred, replicas retain historical information for each update. This information is stored in the target entry's `ds-sync-hist` operational attribute.

Replication resolves the following conflicts automatically using the historical information to order changes correctly:

- The attributes of a given entry are modified concurrently in different ways on different replicas.

- An entry is renamed on one replica while being modified on another replica.

- An entry is renamed on one replica while being renamed in a different way on another replica.

- An entry is deleted on one replica while being modified on another replica.

- An entry is deleted and another entry with the same DN added on one replica while the same entry is being modified on another replica.

Replication cannot resolve the following naming conflicts, so you must resolve them manually:

- Different entries that share the same DN are added concurrently on multiple replicas.

- An entry on one replica is moved (renamed) to use the same DN as a new entry concurrently added on another replica.

- A parent entry is deleted on one replica while a child entry is added or renamed concurrently on another replica.

When replication cannot resolve naming conflicts automatically, the server renames the conflicting entry using its `entryUUID` operational attribute. The resulting conflicting entry has a DN with the following form:

```
entryuuid=entryUUID-value+original-RDN,original-parent-DN
```

For each conflicting entry named in this way, resolve the conflict manually:

1. Get the conflicting entry or entries, and the original entry if available.

    The following example shows the result on one replica of a naming conflict when a `newuser` entry was added concurrently on two replicas:

    ```
    $ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=newuser)"
    dn: uid=newuser,ou=People,dc=example,dc=com
    objectClass: top
    objectClass: inetOrgPerson
    objectClass: organizationalPerson
    objectClass: person
    mail: newuser@example.com
    sn: User
    cn: New User
    ou: People
    description: Added on server 1
    uid: newuser

    dn: entryuuid=2f1b58c3-4bee-4215-88bc-88202a7bcb9d+uid=newuser,ou=People,dc=example,dc=com
    objectClass: top
    objectClass: inetOrgPerson
    objectClass: organizationalPerson
    objectClass: person
    mail: newuser@example.com
    sn: User
    cn: New User
    ou: People
    description: Added on server 2
    uid: newuser
    ```

2. If you want to preserve changes made on the conflicting entry or entries, apply the changes manually.

    The following example shows a modification to preserve both description values:

    ```
    $ cat fix-conflict.ldif
    dn: uid=newuser,ou=People,dc=example,dc=com
    changetype: modify
    add: description
    description: Added on server 2


    $ ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword password fix-conflict.ldif
    ```

    For additional examples demonstrating how to apply changes to directory entries, see "Updating the Directory" in the *Developer's Guide*.

3. After making any necessary changes, manually delete the conflicting entry or entries.

The following example deletes the conflicting entry:

```
$ ldapdelete \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 entryuuid=2f1b58c3-4bee-4215-88bc-88202a7bcb9d+uid=newuser,ou=People,dc=example,dc=com
```

For additional examples, see "Deleting Entries" in the *Developer's Guide*.

Chapter 9
Backing Up and Restoring Data

This chapter covers management of directory data backup archives. For information on managing directory data in an interoperable format that is portable between LDAP directory servers, see "*Managing Directory Data*" instead. In this chapter you will learn to:

- Create backup archives

- Restore data from backup archives

An OpenDJ directory server lets you back up and restore your data either in compressed, binary format, or in LDIF. This chapter shows you how to back up and to restore OpenDJ data from archives, and explains portability of backup archives, and how to back up server configuration data.

> **Important**
>
> As explained in "About Database Backends", cleanup processes applied by database backends can be writing data even when there are no pending client or replication operations. To back up a server using a file system snapshot, you must *stop the server before taking the snapshot*.

9.1. Backing Up Directory Data

A `bak/` directory is created when you install an OpenDJ server as a location to save binary backups. When you create a backup, the `bak/backup.info` file contains information about the archive.

Archives produced by the **backup** command contain backups only of the directory data. Backups of server configuration are found in `config/archived-configs/`.

> **Important**
>
> The **backup** command can encrypt the backup data. It encrypts the data using a symmetric key that is stored with the server configuration. The symmetric key is encrypted in turn with the server's public key that is also stored with the server configuration.
>
> When multiple servers are configured to replicate data as described in "*Managing Data Replication*", the servers replicate the keys as well, allowing any server replica to decrypt the backup data.
>
> *If ever all servers in the replication topology are lost, new servers can no longer decrypt any encrypted backup files.*
>
> To work around this limitation, maintain a file system backup of at least one server from each replication topology in your deployment. To recover from a disaster where all servers in the topology were lost, restore

the server files from the file system backup, and start the restored server. Other new servers whose data you restore from encrypted backup can then obtain the decryption keys from the restored server as described in "To Restore a Replica".

This section includes the following procedures:

• "To Back Up Data Immediately"

• "To Schedule Data Backup"

• "To Schedule Incremental Data Backup"

To Back Up Data Immediately

To perform online backup, you start backup as a task by connecting to the administrative port and authenticating as a user with the `backend-backup` privilege, and also setting a start time for the task by using the `--start` option.

To perform offline backup when an OpenDJ directory server is stopped, you run the **backup** command, described in backup(1) in the *Reference*, without connecting to the server, authenticating, or requesting a backup task.

• Use one of the following alternatives:

    • Back up only the database for Example.com, where the data is stored in the backend named `userRoot`.

       The following example requests an online backup task that starts immediately, backing up only the `userRoot` backend:

```
$ backup \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backendID userRoot \
 --backupDirectory /path/to/opendj/bak \
 --start 0 \
 --trustAll
```

    • Stop the server to back up Example.com data offline.

       The following example stops the OpenDJ server, runs offline backup, and starts the server after backup has completed:

```
$ stop-ds --quiet
$ backup --offline --backendID userRoot --backupDirectory /path/to/opendj/bak
$ start-ds --quiet
```

- Back up all user data on the server.

  The following example requests an online backup task that starts immediately, backing up all backends:

  ```
  $ backup \
   --hostname opendj.example.com \
   --port 4444 \
   --bindDN "cn=Directory Manager" \
   --bindPassword password \
   --backUpAll \
   --backupDirectory /path/to/opendj/bak \
   --start 0 \
   --trustAll
  ```

To Schedule Data Backup

You can schedule online data backup using **crontab** format.

- Back up all user data every night at 2 AM, and notify diradmin@example.com when finished, or on error:

  ```
  $ backup \
   --hostname opendj.example.com \
   --port 4444 \
   --bindDN "cn=Directory Manager" \
   --bindPassword password \
   --backUpAll \
   --backupDirectory /path/to/opendj/bak \
   --recurringTask "00 02 * * *" \
   --completionNotify diradmin@example.com \
   --errorNotify diradmin@example.com \
   --trustAll
  ```

To Schedule Incremental Data Backup

You can schedule an incremental backup by using the `--incremental` option. If you do not set the `--incrementalBaseID` option, then the server increments based on the last backup taken.

- Back up `userRoot` backend data incrementally every night at 3 AM, and notify diradmin@example.com when finished, or on error:

```
$ backup \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backupDirectory /path/to/opendj/bak \
 --backendID userRoot \
 --incremental \
 --recurringTask "00 03 * * *" \
 --completionNotify diradmin@example.com \
 --errorNotify diradmin@example.com \
 --trustAll
```

9.2. Restoring Directory Data From Backup

When you restore data, the procedure to follow depends on whether the directory server is replicated.

To Restore a Standalone Server

To restore directory data when the server is online, you start a restore task by connecting to the administrative port and authenticating as a user with the `backend-restore` privilege. You set a start time for the task with the `--start` option.

To restore data when the server is offline, you run the **restore** command, described in restore(1) in the *Reference*, without connecting to the server, authenticating, or requesting a restore task.

- Use one of the following alternatives:

  - Stop the server to restore data for Example.com.

    The following example stops the server, restores data offline from one of the available backups, and starts the server after the restore is complete:

```
$ stop-ds --quiet
$ restore --offline --backupDirectory /path/to/opendj/bak --listBackups
Backup ID:          <BACKUP_ID1>
Backup Date:        <DATESTAMP1>
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none

Backup ID:          <BACKUP_ID2>
Backup Date:        <DATESTAMP2>
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none
$ restore --offline --backupDirectory /path/to/opendj/bak --backupID ${BACKUP_ID1}
$ start-ds --quiet
```

• Schedule the restore as a task to begin immediately.

  The following example requests an online restore task, scheduled to start immediately:

```
$ restore \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backupDirectory /path/to/opendj/bak \
 --backupID ${BACKUP_ID1} \
 --start 0 \
 --trustAll
```

To Restore a Replica

After you restore a replica from backup, replication brings the replica up to date with changes that happened after you created the backup. In order to bring the replica up to date, replication must apply changes that happened after the backup was made. Replication uses internal change log records to determine what changes to apply.

Internal change log records are not kept forever, though. Replication is configured to purge the change log of old changes, preventing the log from growing indefinitely. Yet, for replication to determine what changes to apply to a restored replica, it must find change log records dating back at least to the last change in the backup. In other words, replication can bring the restored replica up to date *as long as the change log records used to determine which changes to apply have not been purged*.

Therefore, when you restore a replicated server from backup, make sure the backup you use is newer than the last purge of the replication change log (default: 3 days). If all your backups are older than the replication purge delay, do not restore from a backup, but instead initialize a new replica as described in "Initializing Replicas".

1. (Optional) When restoring data from encrypted backup, configure replication between the new replica server and a server from the existing topology as described in "Configuring Replication".

   If the backup is not encrypted, you can skip this step.

   This step initiates the OpenDJ server's key distribution capability, which makes it possible for the replica to obtain secret keys for decrypting backup data from existing replicas. Without the secret key for decryption, the new server cannot read the encrypted backup to restore.

   > **Important**
   >
   > After a disaster leading to the loss of all servers in the replication topology, you must first restore a server from file system backup as described in "Backing Up Directory Data".
   >
   > When the restored server is running, configure replication between the new replica server and the restored server.

   It is not necessary to initialize replication in this step, as you will restore the data in the next step.

2. Restore the server database from the backup archive that you are sure is newer than the last purge of the replication change log.

   For examples of restoring from backup, see "To Restore a Standalone Server".

Chapter 10

Configuring Password Policy

This chapter covers password policy including examples for common use cases. In this chapter you will learn to:

- Decide what type of password policy is needed

- Discover which password policy applies for a given user

- Configure server-based and subentry-based password policies

- Assign password policies to users and to groups

- Configure automated password generation, password storage schemes, and validation of new passwords to reject invalid passwords before they are set

If you want to synchronize password policy across your organization and your applications go to the directory for authentication, then the directory can be a good place to enforce your password policy uniformly. Even if you do not depend on the directory for all your password policy, you no doubt still want to consider directory password policy if only to choose the appropriate password storage scheme.

10.1. About OpenDJ Password Policies

OpenDJ password policies govern not only passwords, but also account lockout, and how OpenDJ servers provide notification about account status.

OpenDJ servers support password policies as part of the server configuration, and subentry password policies as part of the (replicated) user data.

10.1.1. Server-Based Password Policies

You manage server-based password policies in the OpenDJ server configuration by using the **dsconfig** command. As they are part of the server configuration, such password policies are not replicated. You must instead apply password policy configuration updates to each replica in your deployment.

By default, an OpenDJ server includes two password policy configurations, one default for all users, and another default for directory root DN users, such as `cn=Directory Manager`. You can see all the default password policy settings by using the **dsconfig** command as follows:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --advanced \
 --trustAll \
 --no-prompt
Property                                 : Value(s)
-----------------------------------------:-----------------------------------
account-status-notification-handler      : -
allow-expired-password-changes           : false
allow-multiple-password-values           : false
allow-pre-encoded-passwords              : false
allow-user-password-changes              : true
default-password-storage-scheme          : Salted SHA-512
deprecated-password-storage-scheme       : -
expire-passwords-without-warning         : false
force-change-on-add                      : false
force-change-on-reset                    : false
grace-login-count                        : 0
idle-lockout-interval                    : 0 s
java-class                               : org.opends.server.core.PasswordPoli
                                         : cyFactory
last-login-time-attribute                : -
last-login-time-format                   : -
lockout-duration                         : 0 s
lockout-failure-count                    : 0
lockout-failure-expiration-interval      : 0 s
max-password-age                         : 0 s
max-password-reset-age                   : 0 s
min-password-age                         : 0 s
password-attribute                       : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval     : 5 d
password-generator                       : Random Password Generator
password-history-count                   : 0
password-history-duration                : 0 s
password-validator                       : -
previous-last-login-time-format          : -
require-change-by-time                    : -
require-secure-authentication            : false
require-secure-password-changes          : false
skip-validation-for-administrators       : false
state-update-failure-policy              : reactive
```

For detailed descriptions of each property, see the *Server Configuration Reference*.

Notice that many capabilities are not set by default: no lockout, no password expiration, no multiple passwords, no password validator to check that passwords contain the appropriate mix of characters. This means that if you decide to use the directory to enforce password policy, you must configure at least the default password policy to meet your needs.

A few basic protections are configured by default. When you import LDIF with `userPassword` values, An OpenDJ directory server applies a one-way hash to the values before storing them. When a user provides a password value during a bind, for example, the server hashes the value provided and compares it with the stored value. This prevents even the directory manager from recovering the plain text value of a user's password:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userpassword
dn: uid=bjensen,ou=People,dc=example,dc=com
userpassword: {SSHA512}<hash>
```

In addition, users can change their passwords provided that you have granted them access to do so. An OpenDJ directory server uses the `userPassword` attribute to store passwords by default, rather than the `authPassword` attribute, which is designed to store passwords hashed by the client application.

10.1.2. Subentry-Based Password Policies

You manage subentry password policies by adding the subentries alongside the user data. An OpenDJ directory server can therefore replicate subentry password policies. The advantages are that you only need to add them once, and that administrators of user data can edit subentry password policies without having directory administrator access.

Subentry password policies support the Internet-Draft Password Policy for LDAP Directories (version 09). A subentry password policy effectively overrides settings in the default password policy defined in the OpenDJ configuration. Settings not supported or not included in the subentry password policy are inherited from the default password policy.

"Subentry Policy Attributes vs. Server Properties" lists Internet-Draft password policy attributes that override the default password policy when you set them in the subentry:

Subentry Policy Attributes vs. Server Properties

Internet-Draft Policy Attribute	Overrides This Server Policy Property
pwdAllowUserChange	allow-user-password-changes
pwdMustChange	force-change-on-reset
pwdGraceAuthNLimit	grace-login-count
pwdLockoutDuration	lockout-duration
pwdMaxFailure	lockout-failure-count
pwdFailureCountInterval	lockout-failure-expiration-interval
pwdMaxAge	max-password-age

Internet-Draft Policy Attribute	Overrides This Server Policy Property
pwdMinAge	min-password-age
pwdAttribute	password-attribute
pwdSafeModify	password-change-requires-current-password
pwdExpireWarning	password-expiration-warning-interval
pwdInHistory	password-history-count

The following Internet-Draft password policy attributes are not taken into account by OpenDJ servers:

- pwdCheckQuality, because OpenDJ servers have password validators. Set password validators to use in the default password policy, for example.

- pwdMinLength, because this is handled by the length-based password validator. Configure this validator as part of the default password policy, for example.

- pwdLockout, because OpenDJ servers can deduce whether lockout is configured based on the values of other lockout-related password policy attributes.

Values of the following properties are inherited from the default password policy for Internet-Draft based password policies:

- account-status-notification-handlers

- allow-expired-password-changes

- allow-multiple-password-values

- allow-pre-encoded-passwords

- default-password-storage-schemes

- deprecated-password-storage-schemes

- expire-passwords-without-warning

- force-change-on-add

- idle-lockout-interval

- last-login-time-attribute

- last-login-time-format

- max-password-reset-age

- password-generator

- password-history-duration

- password-validators

- previous-last-login-time-formats

- require-change-by-time

- require-secure-authentication

- require-secure-password-changes

- skip-validation-for-administrators

- state-update-failure-policy

If you would rather specify password validators for your policy, you can configure password validators for a subentry password policy by adding the auxiliary object class pwdValidatorPolicy and setting the multi-valued attribute, ds-cfg-password-validator, to the DNs of the password validator configuration entries.

The following example shows a subentry password policy that references two password validator configuration entries. The Character Set password validator determines whether a proposed password is acceptable by checking whether it contains a sufficient number of characters from one or more user-defined character sets and ranges. The length-based password validator determines whether a proposed password is acceptable based on whether the number of characters it contains falls within an acceptable range of values. Both are enabled in the default OpenDJ server configuration:

```
dn: cn=Subentry Password Policy with Validators,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: pwdPolicy
objectClass: pwdValidatorPolicy
cn: Subentry Password Policy with Validators
pwdAttribute: userPassword
pwdLockout: TRUE
pwdMaxFailure: 3
pwdFailureCountInterval: 300
pwdLockoutDuration: 300
pwdAllowUserChange: TRUE
pwdSafeModify: TRUE
ds-cfg-password-validator: cn=Character Set,cn=Password Validators,cn=config
ds-cfg-password-validator: cn=Length-Based Password Validator,
 cn=Password Validators,cn=config
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }
```

If a referenced password validator cannot be found, then the OpenDJ server logs an error message when the password policy is invoked. This can occur, for example, when a subentry password policy is replicated to a directory server where the password validator is not (yet) configured. In that case, when a user attempts to change their password, the server fails to find the referenced password validator.

See also "To Create a Subentry-Based Password Policy".

10.1.3. Which Password Policy Applies

The password policy that applies to a user is identified by the operational attribute, `pwdPolicySubentry`. The default global access control instructions prevent this operational attribute from being visible to normal users. The following example gives access to administrators:

```
$ cat manage-pwp.ldif
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "pwdPolicySubentry||ds-pwp-password-policy-dn")
 (version 3.0;acl "Allow Administrators to manage user's password policy";
 allow (all) (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 manage-pwp.ldif
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 pwdPolicySubentry
dn: uid=bjensen,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Default Password Policy,cn=Password Policies,cn=config
```

10.2. Configuring Password Policies

You configure server-based password policies by using the **dsconfig** command. Notice that server-based password policies are part of the server configuration, and therefore not replicated. Alternatively, you can configure a subset of password policy features by using subentry-based password policies that are stored with the replicated server data.

This section covers both server-based and subentry-based password policies. It includes the following procedures:

- "To Adjust the Default Password Policy"

- "To Configure the Default Policy to Meet NIST Requirements"

- "To Create a Server-Based Password Policy"

- "To Create a Subentry-Based Password Policy"

To Adjust the Default Password Policy

You can reconfigure the default password policy, for example, to check that passwords do not contain complete attribute values, and to prevent password reuse. The default policy is a server-based password policy.

1. Apply the changes to the default password policy:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set password-history-count:7 \
 --set password-validator:Attribute\ Value \
 --trustAll \
 --no-prompt
```

2. Check your work:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --trustAll
Property                                  : Value(s)
------------------------------------------:-------------------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-user-password-changes               : true
default-password-storage-scheme           : Salted SHA-512
deprecated-password-storage-scheme        : -
expire-passwords-without-warning          : false
force-change-on-add                       : false
force-change-on-reset                     : false
grace-login-count                         : 0
idle-lockout-interval                     : 0 s
last-login-time-attribute                 : -
last-login-time-format                    : -
lockout-duration                          : 0 s
lockout-failure-count                     : 0
lockout-failure-expiration-interval       : 0 s
max-password-age                          : 0 s
max-password-reset-age                    : 0 s
min-password-age                          : 0 s
password-attribute                        : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval      : 5 d
password-generator                        : Random Password Generator
password-history-count                    : 7
password-history-duration                 : 0 s
password-validator                        : Attribute Value
```

```
previous-last-login-time-format          : -
require-change-by-time                    : -
require-secure-authentication            : false
require-secure-password-changes          : false
```

3. Test changes to the default password policy.

   For example, the following tests demonstrate how the attribute value password validator works. The attribute value password validator rejects a new password when the password is contained in attribute values on the user's entry.

   By default, the attribute value password validator checks all attributes, checks whether portions of the password string match attribute values, where the portions are strings of length 5, and checks the reverse of the password as well:

```
$ dsconfig \
 get-password-validator-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --validator-name Attribute\ Value \
 --trustAll \
 --no-prompt
Property                : Value(s)
-----------------------:---------
check-substrings        : true
enabled                 : true
match-attribute         : -
min-substring-length    : 5
test-reversed-password  : true
```

   Consider the attributes present on Babs Jensen's entry:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)"
dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: organizationalPerson
objectClass: person
mail: bjensen@example.com
roomNumber: 0209
preferredLanguage: en, ko;q=0.8
manager: uid=trigden, ou=People, dc=example,dc=com
ou: Product Development
ou: People
givenName: Barbara
telephoneNumber: +1 408 555 1862
sn: Jensen
cn: Barbara Jensen
cn: Babs Jensen
homeDirectory: /home/bjensen
facsimileTelephoneNumber: +1 408 555 1992
gidNumber: 1000
uidNumber: 1076
description: Original description
uid: bjensen
l: San Francisco
departmentNumber: 3001
street: 201 Mission Street Suite 2900
```

Using the attribute value password validator, passwords like `bjensen12` and `babsjensenspwd` are not valid because substrings of the password match complete attribute values:

```
$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword hifalutin \
 --newPassword bjensen12
The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry

$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword hifalutin \
 --newPassword babsjensenspwd
The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry
```

The attribute value password validator does not check, however, whether the password contains substrings of attribute values:

```
$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword hifalutin \
 --newPassword babsp4ssw0rd
The LDAP password modify operation was successful

$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword babsp4ssw0rd \
 --newPassword example.com
The LDAP password modify operation was successful
```

To avoid the problem of the latter example, you could use a dictionary password validator where the dictionary includes `example.com`. For an example using a dictionary password validator, see "To Configure the Default Policy to Meet NIST Requirements".

To Configure the Default Policy to Meet NIST Requirements

As described in "To Adjust the Default Password Policy", the default policy is a server-based password policy.

You can change the default password policy to use following rules that are inspired by NIST 800-63 requirements (as of September 2016):

- Use a strong password storage scheme.

  The example in this procedure uses PBKDF2, which requires more processing time than Salted SHA-512 (the default).

- Enforce a minimum password length of 8 characters.

- Check for matches in a dictionary of compromised passwords.

  Use a file in the same format as `config/wordlist.txt`, where each line contains a common password. Lists of common passwords can be found online.

- Do not use composition rules for password validation.

  In other words, do not require a mix of special characters, upper and lower case letters, numbers, or other composition rules.

- Do not enforce arbitrary password changes.

  In other words, do not set a maximum password age.

Follow these steps to make this the default password policy:

1. Create a password validator for a minimum length of 8 characters:

```
$ dsconfig \
 create-password-validator \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --validator-name "At least 8 characters" \
 --type length-based \
 --set enabled:true\
 --set min-password-length:8 \
 --trustAll \
 --no-prompt
```

2. Create a password validator for checking common compromised passwords:

```
# Obtain a copy of a dictionary list of common passwords:
$ cp 10k_most_common.txt /path/to/opendj/config/
$ dsconfig \
 create-password-validator \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --validator-name "Common passwords" \
 --type dictionary \
 --set enabled:true \
 --set dictionary-file:config/10k_most_common.txt \
 --trustAll \
 --no-prompt
```

3. Configure the password policy as the default:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set default-password-storage-scheme:PBKDF2 \
 --set password-validator:"At least 8 characters" \
 --set password-validator:"Common passwords" \
 --trustAll \
 --no-prompt
```

4. Check that the default password policy works appropriately.

   The following example shows a rejected password modification:

```
$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword hifalutin \
 --newPassword secret12
The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry
```

The following example shows an accepted password modification:

```
$ ldappasswordmodify \
 --port 1389 \
 --authzID "dn:uid=bjensen,ou=people,dc=example,dc=com" \
 --currentPassword hifalutin \
 --newPassword aET1OjQeVJECSMgxDPs3U6In
The LDAP password modify operation was successful
```

To Create a Server-Based Password Policy

You can add a password policy, for example, for new users who have not yet used their credentials to bind.

1. Create the new password policy:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "New Account Password Policy" \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set force-change-on-add:true \
 --set password-attribute:userPassword \
 --type password-policy \
 --trustAll \
 --no-prompt
```

2. Check your work:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "New Account Password Policy" \
```

```
 --trustAll
Property                                : Value(s)
----------------------------------------:---------------
account-status-notification-handler     : -
allow-expired-password-changes          : false
allow-user-password-changes             : true
default-password-storage-scheme         : Salted SHA-512
deprecated-password-storage-scheme      : -
expire-passwords-without-warning        : false
force-change-on-add                     : true
force-change-on-reset                   : false
grace-login-count                       : 0
idle-lockout-interval                   : 0 s
last-login-time-attribute               : -
last-login-time-format                  : -
lockout-duration                        : 0 s
lockout-failure-count                   : 0
lockout-failure-expiration-interval     : 0 s
max-password-age                        : 0 s
max-password-reset-age                  : 0 s
min-password-age                        : 0 s
password-attribute                      : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval    : 5 d
password-generator                      : -
password-history-count                  : 0
password-history-duration               : 0 s
password-validator                      : -
previous-last-login-time-format         : -
require-change-by-time                  : -
require-secure-authentication           : false
require-secure-password-changes         : false
```

If you use a password policy like this, then you will want to change the user's policy again when the new user successfully updates the password. For instructions on assigning a server-based password policy, see "To Assign a Password Policy to a User".

To Create a Subentry-Based Password Policy

You can add a subentry to configure a password policy that applies to Directory Administrators.

1. Create the entry that specifies the password policy:

```
$ cat subentry-password-policy.ldif
dn: cn=Subentry Password Policy with Validators,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: pwdPolicy
objectClass: pwdValidatorPolicy
cn: Subentry Password Policy with Validators
pwdAttribute: userPassword
pwdLockout: TRUE
pwdMaxFailure: 3
pwdFailureCountInterval: 300
pwdLockoutDuration: 300
pwdAllowUserChange: TRUE
pwdSafeModify: TRUE
ds-cfg-password-validator: cn=Character Set,cn=Password Validators,cn=config
ds-cfg-password-validator: cn=Length-Based Password Validator,
 cn=Password Validators,cn=config
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }
```

2. Add the policy to the directory:

```
$ ldapmodify \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 subentry-password-policy.ldif
```

3. Check that the policy applies as specified.

In this example, the policy should apply to a Directory Administrator, while a normal user has the default password policy. Here, Kirsten Vaughan is a member of the Directory Administrators group, and Babs Jensen is not a member:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 pwdPolicySubentry
dn: uid=kvaughan,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Subentry Password Policy with Validators,dc=example,dc=com

$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 pwdPolicySubentry
dn: uid=bjensen,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Default Password Policy,cn=Password Policies,cn=config
```

10.3. Assigning Password Policies

You assign subentry-based password policies for a subtree of the DIT by adding the policy to an LDAP subentry whose immediate superior is the root of the subtree. In other words, you can add the subentry-based password policy under ou=People,dc=example,dc=com to have it apply to all entries under ou=People,dc=example,dc=com. You can also use the capabilities of LDAP subentries to refine the scope of application.

You assign server-based password policies by using the ds-pwp-password-policy-dn attribute.

"Server-Based and Subentry Password Policies" compares the types of password policy.

Server-Based and Subentry Password Policies
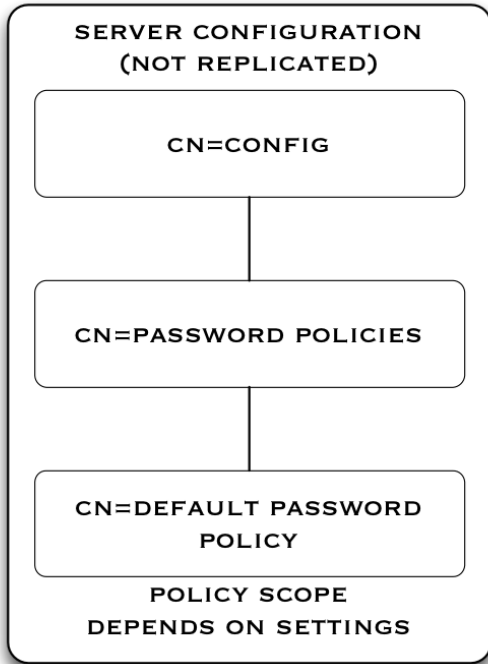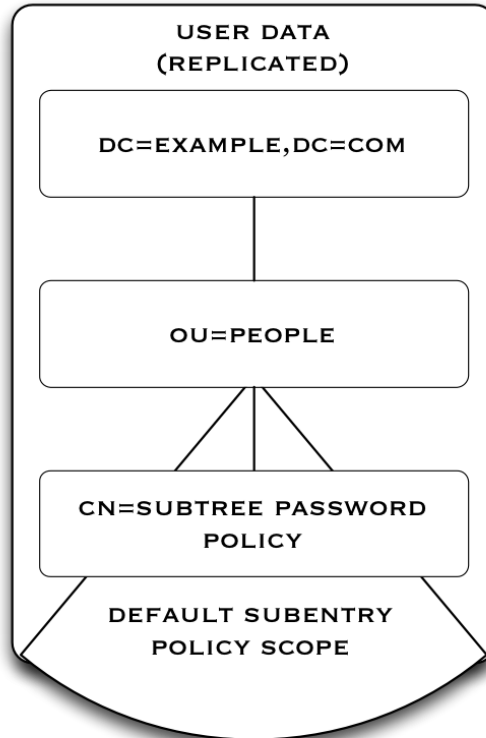
To Assign a Password Policy to a User

1. Give administrators the right to manage users' password policies:

```
$ cat manage-pwp.ldif
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "pwdPolicySubentry||ds-pwp-password-policy-dn")
 (version 3.0;acl "Allow Administrators to manage user's password policy";
 allow (all) (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 manage-pwp.ldif
```

Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Update the user's `ds-pwp-password-policy-dn` attribute:

```
$ cat newuser.ldif
dn: uid=newuser,ou=People,dc=example,dc=com
uid: newuser
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: New User
sn: User
ou: People
mail: newuser@example.com
userPassword: changeme
ds-pwp-password-policy-dn: cn=New Account Password Policy,cn=Password Policies,cn=config

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 newuser.ldif
```

3. Check your work:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=newuser)" \
 pwdPolicySubentry
dn: uid=newuser,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=New Account Password Policy,cn=Password Policies,cn=config
```

To Assign a Password Policy to a Group

You can use a collective attribute to assign a password policy:

1. Give an administrator the privilege to write subentries, such as those used for setting collective attributes:

   ```
   $ cat subentry-write.ldif
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: subentry-write

   $ ldapmodify \
    --port 1389 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    subentry-write.ldif
   ```

   Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Create a subentry defining the collective attribute that sets the `ds-pwp-password-policy-dn` attribute for group members' entries:

```
$ cat pwp-coll.ldif
dn: cn=Password Policy for Dir Admins,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Password Policy for Dir Admins
ds-pwp-password-policy-dn;collective: cn=Root Password Policy,cn=Password Policies,cn=config
subtreeSpecification: { base "ou=People", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)"}

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 pwp-coll.ldif
```

3. Check your work:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 pwdPolicySubentry
dn: uid=kvaughan,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Root Password Policy,cn=Password Policies,cn=config
```

To Assign Password Policy for an Entire Branch

A password policy subentry password policy to assign the policy to the entries under a base DN:

1. Give an administrator the privilege to write subentries, such as those used for setting password policies:

```
$ cat subentry-write.ldif
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 subentry-write.ldif
```

Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the

administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Create a password policy with a `subtreeSpecification` to assign the policy to all entries under a base DN.

   The following example creates a password policy for entries under `ou=People,dc=example,dc=com`:

```
$ cat people-pwp.ldif
dn: cn=People Password Policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: pwdPolicy
cn: People Password Policy
pwdAttribute: userPassword
pwdLockout: TRUE
pwdMaxFailure: 3
pwdFailureCountInterval: 300
pwdLockoutDuration: 300
pwdAllowUserChange: TRUE
pwdSafeModify: TRUE
subtreeSpecification: { base "ou=people" }

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 people-pwp.ldif
```

   Notice the subtree specification used to assign the policy, `{ base "ou=people" }`. You can relax the subtree specification value to `{}` to apply the password policy to all sibling entries (all entries under `dc=example,dc=com`), or further restrict the subtree specification by adding a `specificationFilter`. See "Collective Attributes" in the *Developer's Guide* for more information.

3. Check your work:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=alutz)" \
 pwdPolicySubentry
dn: uid=alutz,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=People Password Policy,dc=example,dc=com
```

   If everything is correctly configured, then the password policy should be assigned to users whose entries are under `ou=People,dc=example,dc=com`.

10.4. Configuring Password Generation

Password generators are used by an OpenDJ server during the LDAP Password Modify extended operation to construct a new password for the user. In other words, a directory administrator resetting a user's password can have an OpenDJ server generate the new password by using the **ldappasswordmodify** command, described in ldappasswordmodify(1) in the *Reference*:

```
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen"
The LDAP password modify operation was successful
Generated Password:   <random>
```

The default password policy shown in "To Adjust the Default Password Policy" uses the Random Password Generator, described in the *Server Configuration Reference*:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --property password-generator \
 --trustAll \
 --no-prompt
Property           : Value(s)
-------------------:--------------------------
password-generator : Random Password Generator
$ dsconfig \
 get-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --property password-generator \
 --trustAll \
 --no-prompt
Property           : Value(s)
-------------------:--------------------------
password-generator : Random Password Generator
```

Notice that the default configuration for the Random Password Generator defines two `password-character-set` values, and then uses those definitions in the `password-format` so that generated passwords have eight characters: three from the `alpha` set, followed by two from the `numeric` set, followed by three from the `alpha` set. The `password-character-set` name must be ASCII.

To set the password generator that the OpenDJ server employs when constructing a new password for a user, set the `password-generator` property for the password policy that applies to the user.

The following example does not change the password policy, but instead changes the Random Password Generator configuration, and then demonstrates a password being generated upon reset:

```
$ dsconfig \
 set-password-generator-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --generator-name "Random Password Generator" \
 --remove password-character-set:alpha:abcdefghijklmnopqrstuvwxyz \
 --add password-character-set:alpha:ABCDEFGHIJKLMNOPQRSTUVWabcdefghijklmnopqrstuvwxyz \
 --add password-character-set:"punct:,./\`!@#\$%^&*:\;[]\"\'\(\)+=-_~\\" \
 --set password-format:alpha:3,punct:1,numeric:2,punct:2,numeric:3,alpha:3,punct:2 \
 --trustAll \
 --no-prompt
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen"
The LDAP password modify operation was successful
Generated Password:  <random>
```

If you also set up a password validator in the password policy as shown in "To Adjust the Default Password Policy" and further described in "Configuring Password Validation", make sure the generated passwords are acceptable to the validator.

10.5. Configuring Password Storage

Password storage schemes, described in the *Server Configuration Reference*, encode new passwords provided by users so that they are stored in an encoded manner. This makes it difficult or impossible to determine the cleartext passwords from the encoded values. Password storage schemes also determine whether a cleartext password provided by a client matches the encoded value stored by the server.

OpenDJ servers offer a variety of reversible and one-way password storage schemes. With a reversible encryption scheme, an attacker who gains access to the server can recover the cleartext passwords. With a one-way hash storage scheme, the attacker who gains access to the server must still crack the password by brute force, encoding passwords over and over to generate guesses until a match is found. If you have a choice, use a one-way password storage scheme.

Some one-way hash functions are not designed specifically for password storage, but also for use in message authentication and digital signatures. Such functions, like those defined in the Secure Hash Algorithm (SHA-1 and SHA-2) standards, are designed for high performance. Because they are fast, they allow the server to perform authentication at high throughput with low response times. However, high-performance algorithms also help attackers use brute force techniques. One estimate in 2017 is that a single GPU can calculate over one billion SHA-512 hashes per second.

> **Warning**
>
> Some one-way hash functions are designed to be computationally *expensive*. Such functions, like PBKDF2 and Bcrypt, are designed to be relatively slow even on modern hardware. This makes them generally less susceptible to brute force attacks. *However*, computationally expensive functions reduce authentication throughput and increase response times. With the default number of iterations, the GPU mentioned above might only calculate 100,000 PBKDF2 hashes per second (or 0.01% of the corresponding hashes calculated with SHA-512). If you use these functions, be aware of the potentially dramatic performance impact and plan your deployment accordingly. Do not use functions like Bcrypt for any accounts that are used for frequent, short-lived connections.

```
$ dsconfig \
 list-password-storage-schemes \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll
Password Storage Scheme : Type          : enabled
------------------------:---------------:--------
3DES                    : triple-des    : true
AES                     : aes           : true
Base64                  : base64        : true
Bcrypt                  : bcrypt        : true
Blowfish                : blowfish      : true
Clear                   : clear         : true
CRYPT                   : crypt         : true
MD5                     : md5           : true
PBKDF2                  : pbkdf2        : true
PKCS5S2                 : pkcs5s2       : true
RC4                     : rc4           : true
Salted MD5              : salted-md5    : true
Salted SHA-1            : salted-sha1   : true
Salted SHA-256          : salted-sha256 : true
Salted SHA-384          : salted-sha384 : true
Salted SHA-512          : salted-sha512 : true
SHA-1                   : sha1          : true
```

As shown in "To Adjust the Default Password Policy", the default password storage scheme for users is Salted SHA-512. When you add users or import user entries with `userPassword` values in cleartext, an OpenDJ server hashes them with the default password storage scheme. Root DN users have a different password policy by default, shown in "To Assign a Password Policy to a Group". The Root Password Policy uses PBKDF2 by default.

> **Tip**
>
> The choice of default password storage scheme for normal users can significantly impact server performance. Each time a normal user authenticates using simple bind (username/password) credentials, the directory server encodes the user's password according to the storage scheme in order to compare it with the encoded value in the user's entry.

Schemes such as Salted SHA-512 call for relatively high-performance encoding. Schemes such as PBKDF2, which are designed to make the encoding process computationally expensive, reduce the bind throughput that can be achieved on equivalent hardware.

Take this performance impact into consideration when choosing a password storage scheme. If you opt for a scheme such as PBKDF2, make sure the directory service has enough compute power to absorb the additional load.

The password storage schemes listed in "Additional Password Storage Scheme Settings" have additional configuration settings.

Additional Password Storage Scheme Settings

Scheme	Setting	Description
Bcrypt	`bcrypt-cost`	The cost parameter specifies a key expansion iteration count as a power of two. A default value of 12 (2^{12} iterations) is considered in 2016 as a reasonable balance between responsiveness and security for regular users.
Crypt	`crypt-password-storage-encryption -algorithm`	Specifies the crypt algorithm to use to encrypt new passwords. The following values are supported: **unix** The password is encrypted with the weak Unix crypt algorithm. This is the default setting. **md5** The password is encrypted with the BSD MD5 algorithm and has a `$1$` prefix. **sha256** The password is encrypted with the SHA256 algorithm and has a `$5$` prefix. **sha512** The password is encrypted with the SHA512 algorithm and has a `$6$` prefix.
PBKDF2	`pbkdf2-iterations`	The number of algorithm iterations. NIST recommends at least 1000. The default is 10000.

You change the default password policy storage scheme for users by changing the applicable password policy, as shown in the following example:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set default-password-storage-scheme:pbkdf2 \
 --trustAll \
 --no-prompt
```

Notice that the change in default password storage scheme does not cause the OpenDJ server to update any stored password values. By default, the server only stores a password with the new storage scheme the next time that the password is changed.

OpenDJ servers prefix passwords with the scheme used to encode them, which means it is straightforward to see which password storage scheme is used. After the default password storage scheme is changed to PBKDF2, old user passwords remain encoded with Salted SHA-512:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userPassword
dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {SSHA512}<hash>
```

When the password is changed, the new default password storage scheme takes effect, as shown in the following example:

```
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword changeit
The LDAP password modify operation was successful
$ ldapsearch \
 --port 1389 \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword changeit \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userPassword
dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2}10000:<hash>
```

When you change the password storage scheme for users, realize that the user passwords must change in order for the OpenDJ server to encode them with the chosen storage scheme. If you are changing the storage scheme because the old scheme was too weak, then you no doubt want users to change their passwords anyway.

If, however, the storage scheme change is not related to vulnerability, you can use the `deprecated-password-storage-scheme` property of the password policy to have the OpenDJ server store the password in the new format after successful authentication. This makes it possible to do password migration for active users without forcing users to change their passwords:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 userPassword
dn: uid=kvaughan,ou=People,dc=example,dc=com
userPassword: {SSHA512}<hash>
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set deprecated-password-storage-scheme:"Salted SHA-1" \
 --trustAll \
 --no-prompt
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 userPassword
dn: uid=kvaughan,ou=People,dc=example,dc=com
userPassword: {PBKDF2}10000:<hash>
```

Notice that with `deprecated-password-storage-scheme` set appropriately, Kirsten Vaughan's password was hashed again after she authenticated successfully.

10.6. Configuring Password Validation

Password validators, described in the *Server Configuration Reference*, are responsible for determining whether a proposed password is acceptable for use. Validators can run checks like ensuring that the password meets minimum length requirements, that it has an appropriate range of characters, or that it is not in the history of recently used passwords.

The following example lists password validators, including validators created in earlier examples:

```
$ dsconfig \
 list-password-validators \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt
Password Validator                   : Type                 : enabled
-------------------------------------:----------------------:--------
At least 8 characters                : length-based         : true
Attribute Value                      : attribute-value      : true
Character Set                        : character-set        : true
Common passwords                     : dictionary           : true
Dictionary                           : dictionary           : false
Length-Based Password Validator      : length-based         : true
Repeated Characters                  : repeated-characters  : true
Similarity-Based Password Validator  : similarity-based     : true
Unique Characters                    : unique-characters    : true
```

The password policy for a user specifies the set of password validators that should be used whenever that user provides a new password. By default, no password validators are configured. You can see an example setting the Default Password Policy to use the Dictionary validator in "To Adjust the Default Password Policy". The following example shows how to set up a custom password validator and assign it to the default password policy.

The custom password validator ensures passwords meet at least three of the following four criteria. Passwords are composed of the following:

• English lowercase characters (a through z)

• English uppercase characters (A through Z)

• Base 10 digits (0 through 9)

• Non-alphabetic characters (for example, !, $, #, %)

Notice how the `character-set` values are constructed. The initial `0:` means the set is optional, whereas `1:` means the set is required:

```
$ dsconfig \
 create-password-validator \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --validator-name "Custom Character Set Password Validator" \
 --set allow-unclassified-characters:true \
 --set enabled:true \
 --set character-set:0:abcdefghijklmnopqrstuvwxyz \
 --set character-set:0:ABCDEFGHIJKLMNOPQRSTUVWXYZ \
 --set character-set:0:0123456789 \
 --set character-set:0:\!\"#\$%\&\'\(\)\*+\,-./:\;\\\<=\>\?@\[\]\^_\`\{\|\}~ \
```

```
  --set min-character-sets:3 \
  --type character-set \
  --trustAll \
  --no-prompt
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set password-validator:"Custom Character Set Password Validator" \
 --trustAll \
 --no-prompt
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword '!ABcd$%^'
```

In the preceding example, the character set of ASCII punctuation, `\!\"#\$%\&\'\(\)\*+\,-./:\;\\\<=\>\?` `@\[\]\^_\`\{\|\}~`, is hard to read because of all of the escape characters. It is equivalent to `!"#$%&'()*` `+,-./:;\<=>?@[]^_`{|}~`. To enter sequences sequences that are difficult to escape properly, use the **dsconfig** in interactive mode, and let it do the escaping for you. You can also use the `--commandFilePath` `{path}` option to save the result of your interactive session to a file for later use in scripts.

An attempt to set an invalid password fails as shown in the following example:

```
$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword hifalutin
The LDAP password modify operation failed with result code 19
Error Message:  The provided new password failed the validation checks defined
in the server: The provided password did not contain characters from at least
3 of the following character sets or ranges: '0123456789',
'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz',
'!"#$%&'()*+,-./:;\<=>?@[]^_`{|}~'
```

Validation does not affect existing passwords, but only takes effect when the password is updated.

You can reference password validators from subentry password policies. See "Subentry-Based Password Policies" for an example.

10.7. Sample Password Policies

The sample password policies in this section demonstrate OpenDJ server-based password policies for several common cases:

- "Enforce Regular Password Changes"

- "Track Last Login Time"

- "Deprecate a Password Storage Scheme"

- "Lock Idle Accounts"

- "Allow Grace Log In to Change Expired Password"

- "Require Password Change on Add or Reset"

Enforce Regular Password Changes

The following commands configure an OpenDJ server-based password policy that sets age limits on passwords, requiring that they change periodically. It also sets the number of passwords to keep in the password history of the entry, thereby preventing users from reusing the same password on consecutive changes:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Enforce Regular Password Changes" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set max-password-age:13w \
 --set min-password-age:4w \
 --set password-history-count:7 \
 --trustAll \
 --no-prompt
```

See also "Assigning Password Policies" for instructions on using the policy.

Track Last Login Time

The following commands configure an OpenDJ server-based password policy that keeps track of the last successful login.

First, set up an attribute to which the OpenDJ directory server can write a timestamp value on successful login.

The following example defines a `lastLoginTime` attribute. Notice that the attribute has the property `USAGE directoryOperation`, meaning this is an operational attribute. When checking schema compliance, the server skips operational attributes. Operational attributes can therefore be added to an entry without changing the entry's object classes. Furthermore, as described in "About Data In LDAP Directories", operational attributes hold information used by the directory itself. Operational attributes are only returned when explicitly requested, and not intended for modification by external

applications. By defining the `lastLoginTime` attribute as operational, you limit its visibility and help prevent client applications from changing its value unless specifically granted access to do so.

For additional information, also see "Search: Listing Active Accounts" in the *Developer's Guide*:

```
$ cat lastLoginTime.ldif
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( lastLoginTime-oid
  NAME 'lastLoginTime'
  DESC 'Last time the user logged in'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation
  X-ORIGIN 'OpenDJ example documentation' )

$ ldapmodify \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 lastLoginTime.ldif
```

Next, create the password policy that causes the OpenDJ directory server to write the timestamp to the attribute on successful login:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Track Last Login Time" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set last-login-time-attribute:lastLoginTime \
 --set last-login-time-format:"yyyyMMddHH'Z'" \
 --trustAll \
 --no-prompt
```

See "Assigning Password Policies" for instructions on using the policy.

Deprecate a Password Storage Scheme

The following commands configure an OpenDJ server-based password policy that you can use when deprecating a password storage scheme. This policy uses elements from "Enforce Regular Password Changes". The OpenDJ server applies the new password storage scheme to hash or to encrypt

passwords when a password changes, or when the user successfully binds with the correct password and the password is currently hashed with a deprecated scheme:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Deprecate a Password Storage Scheme" \
 --type password-policy \
 --set deprecated-password-storage-scheme:Crypt \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set max-password-age:13w \
 --set min-password-age:4w \
 --set password-history-count:7 \
 --trustAll \
 --no-prompt
```

See also "Assigning Password Policies" for instructions on using the policy.

Lock Idle Accounts

The following commands configure an OpenDJ server-based password policy that locks idle accounts. This policy extends the example from "Track Last Login Time" as the OpenDJ server must track last successful login time in order to calculate how long the account has been idle. You must first add the lastLoginTime attribute type in order for an OpenDJ server to accept this new password policy:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Lock Idle Accounts" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set last-login-time-attribute:lastLoginTime \
 --set last-login-time-format:"yyyyMMddHH'Z'" \
 --set idle-lockout-interval:13w \
 --trustAll \
 --no-prompt
```

See also "Assigning Password Policies", and "Configuring Account Lockout".

Allow Grace Log In to Change Expired Password

The following commands configure an OpenDJ server-based password policy that allows users to log in after their password has expired in order to choose a new password:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Allow Grace Login" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set grace-login-count:2 \
 --trustAll \
 --no-prompt
```

See also "Assigning Password Policies" for instructions on using the policy.

Require Password Change on Add or Reset

The following commands configure an OpenDJ server-based password policy that requires new users to change their password after logging in for the first time, and also requires users to change their password after their password is reset:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Require Password Change on Add or Reset" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-512" \
 --set password-attribute:userPassword \
 --set force-change-on-add:true \
 --set force-change-on-reset:true \
 --trustAll \
 --no-prompt
```

See also "Assigning Password Policies" for instructions on using the policy.

Chapter 11

Implementing Account Lockout and Notification

This chapter covers configuration of account lockout and account status notification. In this chapter you will learn to:

- Configure password policies to manage account lockout automatically

- Manage lockout with the **manage-account** command

- Set up email notification of account status

OpenDJ servers support automatic account lockout. The aim of account lockout is not to punish users who mistype their passwords, but instead to protect the directory against attacks in which the attacker attempts to guess a user password, repeatedly attempting to bind until success is achieved.

Account lockout disables a user account after a specified number of successive authentication failures. When you implement account lockout, you can opt to have the OpenDJ server unlock the account after a specified interval, or you can leave the account locked until the password is reset.

> **Note**
>
> You configure account lockout as part of password policy. the server locks an account after the specified number of consecutive authentication failures. Account lockout is not transactional across a replication topology. Under normal circumstances, replication propagates lockout quickly. If replication is ever delayed, an attacker with direct access to multiple replicas could try to authenticate up to the specified number of times on each replica before being locked out on all replicas.

This chapter shows you how to set up account lockout policies by using the **dsconfig** command, described in dsconfig(1) in the *Reference*, and how to intervene manually to lock and unlock accounts by using the **manage-account** command, described in manage-account(1) in the *Reference*.

11.1. Configuring Account Lockout

Account lockout is configured as part of password policy. This section demonstrates configuring account lockout as part of the default password policy. Users are allowed three consecutive failures before being locked out for five minutes. Failures themselves also expire after five minutes.

Change the default password policy to activate lockout using the **dsconfig** command. As the password policy is part of the server configuration, you must manually apply the changes to each replica in a replication topology:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set lockout-failure-count:3 \
 --set lockout-duration:5m \
 --set lockout-failure-expiration-interval:5m \
 --trustAll \
 --no-prompt
```

Users having the default password policy are then locked out after three failed attempts in succession:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail
dn: uid=bjensen,ou=People,dc=example,dc=com
mail: bjensen@example.com
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail
The LDAP bind request failed: 49 (Invalid Credentials)
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail
The LDAP bind request failed: 49 (Invalid Credentials)
$ ldapsearch \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail
The LDAP bind request failed: 49 (Invalid Credentials)
$ ldapsearch \
 --port 1389 \
```

```
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail
The LDAP bind request failed: 49 (Invalid Credentials)
```

11.2. Managing Accounts Manually

This section covers disabling and enabling accounts by using the **manage-account** command. Password reset is covered in the chapter on performing LDAP operations.

To Disable an Account

1. Make sure the user running the **manage-account** command has access to perform the appropriate operations.

    Kirsten Vaughan is a member of the Directory Administrators group. For this example, she must have the `password-reset` privilege, and access to edit user attributes and operational attributes:

    ```
    $ cat manage-account-access.ldif
    dn: uid=kvaughan,ou=People,dc=example,dc=com
    changetype: modify
    add: ds-privilege-name
    ds-privilege-name: password-reset

    dn: ou=People,dc=example,dc=com
    changetype: modify
    add: aci
    aci: (target="ldap:///ou=People,dc=example,dc=com")(targetattr ="*||+")
     (version 3.0;acl "Admins can run amok"; allow(all)
      groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)

    $ ldapmodify \
     --port 1389 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password \
     manage-account-access.ldif
    ```

    Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Set the account status to disabled with the **manage-account** command:

```
$ manage-account \
 set-account-is-disabled \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --operationValue true \
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --trustAll
Account Is Disabled:  true
```

To Activate a Disabled Account

• Clear the disabled status using the **manage-account** command:

```
$ manage-account \
 set-account-is-disabled \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --operationValue false \
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --trustAll
Account Is Disabled:  false
```

11.3. Managing Account Status Notification

OpenDJ can send mail about account status changes. OpenDJ needs an SMTP server to send messages, and needs templates for the mail it sends. By default, message templates are in English, under `/path/to/opendj/config/messages/`.

OpenDJ generates notifications only when OpenDJ writes to an entry or evaluates a user entry for authentication. OpenDJ generates account enabled and account disabled notifications when the user account is enabled or disabled with the **manage-account** command, which writes to the entry. OpenDJ generates password expiration notifications when a user tries to bind.

For example, if you set up an OpenDJ server to send a notification about password expiration, that notification gets triggered when the user authenticates during the password expiration warning interval. The OpenDJ server does not automatically scan entries to send password expiry notifications. OpenDJ servers do implement controls that you can pass in an LDAP search to determine whether a user's password is about to expire. See "*LDAP Controls*" in the *Reference* for a list. You can send notifications based on the results of your search.

To Mail Users About Account Status

The following steps demonstrate how to set up notifications. Whether OpenDJ sends notifications depends on the settings in the password policy, and on account activity as described above.

1. Identify the SMTP server to which OpenDJ sends messages:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set smtp-server:opendj.example.com:25 \
 --trustAll \
 --no-prompt
```

2. Set up OpenDJ to be able to mail users about account status.

   The following example configures OpenDJ to send text-format mail messages:

```
$ dsconfig \
 set-account-status-notification-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SMTP Handler" \
 --set enabled:true \
 --set email-address-attribute-type:mail \
 --trustAll \
 --no-prompt
```

   Notice that OpenDJ finds the user's mail address on the attribute on the user's entry, specified by `email-address-attribute-type`.

   You can also configure the `message-subject` and `message-template-file` properties. Try interactive mode if you plan to do so.

   You find templates for messages by default under the `config/messages` directory. You can edit the templates to suit your purposes.

   If you edit the templates to send HTML rather than text messages, then set the advanced property, `send-email-as-html`, as shown in the following example:

```
$ dsconfig \
 set-account-status-notification-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SMTP Handler" \
 --set enabled:true \
 --set send-email-as-html:true \
 --trustAll \
 --no-prompt
```

3. Adjust applicable password policies to use the account status notification handler you configured:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set account-status-notification-handler:"SMTP Handler" \
 --trustAll \
 --no-prompt
```

About Notification Message Templates

When editing the `config/messages` templates to suit your purposes, you can use the following tokens to have OpenDJ update the message text dynamically.

%%notification-type%%

This token is replaced with the name of the account status notification type for the notification.

%%notification-message%%

This token is replaced with the message for the account status notification.

%%notification-user-dn%%

This token is replaced with the string representation of the DN for the user who is the target of the account status notification.

%%notification-user-attr:*attrname*%%

This token is replaced with the value of the attribute specified by *attrname* from the user's entry. If the specified attribute has multiple values, then OpenDJ uses the first value encountered. If the specified attribute does not have any values, then OpenDJ replaces it with an emtpy string.

%%notification-property:*propname*%%

This token is replaced with the value of the specified notification property from the account status notification. If the specified property has multiple values, then OpenDJ uses the first value encountered. If the specified property does not have any values, then OpenDJ replaces it with an empty string. Valid *propname* values include the following:

- account-unlock-time

- new-password

- old-password

- password-expiration-time

- password-policy-dn

- seconds-until-expiration

- seconds-until-unlock

- time-until-expiration

- time-until-unlock

Chapter 12
Setting Resource Limits

This chapter shows you how to set resource limits that prevent directory clients from using an unfair share of system resources. In this chapter you will learn to:

- Limit the resources devoted to directory searches

- Limit concurrent client connections

- Limit how long connections can remain idle before they are dropped

- Limit the size of client requests

- Understand how resource limits are set for proxied authorization

12.1. Limiting Search Resources

Well-written directory client applications limit the search scope with filters that narrow the number of results returned. This prevents them from performing unindexed searches. By default, an OpenDJ server only allows users with appropriate privileges to perform unindexed searches.

In addition to letting the server prevent unindexed searches, you can set limits on search operations, such as the following:

- The *lookthrough limit* defines the maximum number of candidate entries that an OpenDJ directory server considers when processing a search.

  The default lookthrough limit of 5000 is set by the global server property `lookthrough-limit`.

  You can override the limit per user with the operational attribute, `ds-rlim-lookthrough-limit`.

- The *size limit* sets the maximum number of entries returned for a search.

  The default size limit of 1000 is set by the global server property `size-limit`.

  You can override the limit per user with the operational attribute, `ds-rlim-size-limit`.

  In addition, search requests themselves can include a size limit setting. The **ldapsearch** command has an `--sizeLimit` option.

- The *time limit* defines the maximum processing time OpenDJ devotes to a search operation.

  The default time limit of 1 minute is set by the global server property `time-limit`.

  You can override the limit on a per user basis with the operational attribute, `ds-rlim-time-limit`. Times for `ds-rlim-time-limit` are expressed in seconds.

  In addition, search requests themselves can include a time limit setting. The **ldapsearch** command has an `--timeLimit` option.

- The *idle time limit* defines how long OpenDJ allows idle connections to remain open.

  No default idle time limit is set. You can set an idle time limit by using the global server property `idle-time-limit`.

  You can override the limit on a per user basis with the operational attribute, `ds-rlim-idle-time-limit`. Times for `ds-rlim-idle-time-limit` are expressed in seconds.

- The maximum number of persistent searches is set by the global server property `max-psearches`.

This section includes the following procedures:

- "To Set Search Limits For a User"

- "To Set Search Limits For Users in a Group"

- "To Limit Concurrent Persistent Searches"

To Set Search Limits For a User

1. Give an administrator access to update the operational attributes related to search limits:

```
$ cat search-limits.ldif
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "ds-rlim-lookthrough-limit||ds-rlim-time-limit||ds-rlim-size-limit")
 (version 3.0;acl "Allow Kirsten Vaughan to manage search limits";
 allow (all) (userdn = "ldap:///uid=kvaughan,ou=People,dc=example,dc=com");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 search-limits.ldif
```

2. Change the user entry to set the limits to override:

```
$ cat size-limit-bjensen.ldif
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: ds-rlim-size-limit
ds-rlim-size-limit: 10

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 size-limit-bjensen.ldif
```

When Babs Jensen performs a search returning more than 10 entries, she sees the following message:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(&)"
...
# The LDAP search request failed: 4 (Size Limit Exceeded)
# Additional Information:  This search operation has sent the maximum of 10 entries to the client
```

To Set Search Limits For Users in a Group

1. Give an administrator the privilege to write subentries, such as those used for setting collective attributes:

```
$ cat subentry-write.ldif
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 subentry-write.ldif
```

Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Create an LDAP subentry to specify the limits using collective attributes:

```
$ cat size-limit-collective.ldif
dn: cn=Remove Administrator Search Limits,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Remove Administrator Search Limits
ds-rlim-lookthrough-limit;collective: 0
ds-rlim-size-limit;collective: 0
ds-rlim-time-limit;collective: 0
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 size-limit-collective.ldif
```

3. Check the results:

```
$ ldapsearch \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN uid=kvaughan,ou=people,dc=example,dc=com \
 --searchScope base \
 "(&)" \
 ds-rlim-lookthrough-limit ds-rlim-time-limit ds-rlim-size-limit
ds-rlim-lookthrough-limit: 0
ds-rlim-time-limit: 0
ds-rlim-size-limit: 0
```

To Limit Concurrent Persistent Searches

An LDAP persistent search uses server resources in the following way. Each persistent search opens a connection and keeps it open, though the connection can be idle for long periods of time. When a modification changes data in the search scope, the server returns a search result to the persistent search client. The more concurrent persistent searches, the more work the server has to do for each modification:

*   Set the global property `max-psearches` to limit the total number of concurrent persistent searches that an OpenDJ server accepts.

    The following example sets the limit to 30:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set max-psearches:30 \
 --trustAll \
 --no-prompt
```

12.2. Limiting Concurrent Client Connections

Each connection uses memory. On UNIX and Linux systems, each connection uses an available file descriptor.

You can use the global setting `max-allowed-client-connections` to limit the total number of concurrent client connections that an OpenDJ server accepts. The following example sets the limit to 64K, which is the minimum number of file descriptors that should be available to the OpenDJ server:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set max-allowed-client-connections:65536 \
 --trustAll \
 --no-prompt
```

12.3. Limiting Idle Time

If some client applications leave connections idle for long periods, an OpenDJ server can end up devoting resources to maintaining connections that are no longer used. If your network does not drop such connections eventually, you can configure the server to drop them by setting the global configuration property `idle-time-limit`. By default, no idle time limit is set.

If your network load balancer is configured to drop connections that have been idle for some time, set the OpenDJ idle time limit to a lower value than the idle time limit for the load balancer. This helps to ensure that idle connections are shut down in orderly fashion. Setting the OpenDJ limit lower than the load balancer limit is particularly useful with load balancers that drop idle connections without cleanly closing the connection and notifying the client and server.

> **Note**
>
> OpenDJ servers do not enforce idle timeout for persistent searches.

The following example sets the idle-time-limit to 24 hours:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set idle-time-limit:24h \
 --trustAll \
 --no-prompt
```

12.4. Limiting Maximum Request Size

The default maximum request size of 5 MB is set using the advanced connection handler property `max-request-size`. This is sufficient for most deployments. In cases where clients add groups with large numbers of members, however, some add requests can exceed the 5 MB limit.

The following example increases the limit to 20 MB for the LDAP connection handler:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set max-request-size:20mb \
 --trustAll \
 --no-prompt
```

Note that this setting affects only the size of requests, not responses.

12.5. Resource Limits and Proxied Authorization

Proxied authorization uses a standard LDAP control to permit an application to bind as one user and then carry out LDAP operations on behalf of other users.

When using proxied authorization as described in "Configuring Proxied Authorization" in the *Developer's Guide*, be aware that the resource limits do not change when the user proxies as another user. In other words, resource limits depend on the bind DN, not the proxy authorization identity.

Chapter 13
Implementing Attribute Value Uniqueness

This chapter shows you how to enforce that specified attributes do not have repeated values in different directory entries. You can use attribute uniqueness, for example, to prevent two user entries sharing the same email address. In this chapter you will learn to:

• Enforce uniqueness for user IDs and other attributes

• Limit the scope of attribute value uniqueness

• Manage attribute value uniqueness across replicated directory servers

Some attribute values ought to remain unique. If you are using `uid` values as RDNs to distinguish between millions of user entries stored under `ou=People`, then you do not want your directory to contain two or more identical `uid` values. If your credit card or mobile number is stored as an attribute value on your directory entry, you certainly do not want to share that credit card or mobile number with another customer. The same is true for your email address.

The difficulty for you as directory administrator lies in implementing attribute value uniqueness without sacrificing the high availability that comes from using the servers's loosely consistent, multi-master data replication. Indeed, the OpenDJ replication model lets you maintain write access during network outages for directory applications. Yet, write access during a network outage can result in the same, theoretically unique attribute value getting assigned to two different entries at once. You do not notice the duplicate assignment until the network outage ends and replication resumes.

This chapter shows you how to set up attribute value uniqueness in your directory environment with the following procedures:

• "To Enable Unique UIDs"

• "To Enable Unique Values For Other Attributes"

• "To Limit The Scope of Uniqueness"

• "To Ensure Unique Attribute Values With Replication"

OpenDJ directory server uses the unique attribute plugin to handle attribute value uniqueness. As shown in the examples in this chapter, you can configure the unique attribute plugin to handle one or more attributes and to handle entries under one or more base DNs. You can also configure multiple instances of the plugin for the same OpenDJ directory server.

To Enable Unique UIDs

OpenDJ servers provide a unique attribute plugin that you configure with the **dsconfig** command. By default, the plugin is prepared to ensure attribute values are unique for `uid` attributes.

1. Set the base DN where `uid` should have unique values, and enable the plugin:

```
$ dsconfig \
 set-plugin-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "UID Unique Attribute" \
 --set base-dn:ou=people,dc=example,dc=com \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

   Alternatively, you can specify multiple base DNs for unique values across multiple suffixes:

```
$ dsconfig \
 set-plugin-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "UID Unique Attribute" \
 --set enabled:true \
 --add base-dn:ou=people,dc=example,dc=com \
 --add base-dn:ou=people,dc=example,dc=org \
 --trustAll \
 --no-prompt
```

2. Check that the plugin is working correctly:

```
$ cat change-ajensen.ldif
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
add: uid
uid: bjensen

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 change-ajensen.ldif
# Processing MODIFY request for uid=ajensen,ou=People,dc=example,dc=com
# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value bjensen
 already exists in entry uid=bjensen,ou=People,dc=example,dc=com
```

If you have set up multiple suffixes, you can try something like this:

```
$ cat bjensen-org.ldif
dn: uid=bjensen,ou=People,dc=example,dc=org
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Babs
sn: Jensen
uid: bjensen

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 bjensen-org.ldif
# Processing ADD request for uid=bjensen,ou=People,dc=example,dc=org
# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value bjensen
 already exists in entry uid=bjensen,ou=People,dc=example,dc=com
```

To Enable Unique Values For Other Attributes

You can also configure the unique attribute plugin for use with other attributes, such as `mail`, `mobile`, or attributes you define, for example `cardNumber`.

1. Before you set up the plugin, index the attribute for equality.

    See "Configuring and Rebuilding Indexes" for instructions.

2. Set up the plugin configuration for your attribute.

    You can either add the attribute to an existing plugin configuration, or create a new plugin configuration including the attribute.

When choosing between alternatives, be aware that values must be unique across the attributes and base DNs specified in each plugin configuration. Only group attributes in the same configuration if you want each value to be unique for all attributes. For example, you might create a single plugin configuration for telephone, fax, mobile, and pager numbers. As an alternative example, suppose user IDs are numeric, that user entries also specify `uidNumber`, and that user IDs are normally the same as their `uidNumber`s. In that case, create separate unique attribute configurations for `uid` and `uidNumber`.

Apply one of these alternatives:

- If you want to add the attribute to an existing plugin configuration, do so as shown in the following example which uses the plugin configuration from "To Enable Unique UIDs":

```
$ dsconfig \
 set-plugin-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "UID Unique Attribute" \
 --add type:telephoneNumber \
 --trustAll \
 --no-prompt
```

- If you want to create a new plugin configuration, do so as shown in the following example:

```
$ dsconfig \
 create-plugin \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "Unique phone numbers" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:ou=people,dc=example,dc=com \
 --set type:telephoneNumber \
 --trustAll \
 --no-prompt
```

3. Check that the plugin is working correctly:

```
$ cat telephone-numbers.ldif
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 828 555 1212

dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 828 555 1212

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 telephone-numbers.ldif
# Processing MODIFY request for uid=ajensen,ou=People,dc=example,dc=com
# MODIFY operation successful for DN uid=ajensen,ou=People,dc=example,dc=com
# Processing MODIFY request for uid=bjensen,ou=People,dc=example,dc=com
# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute telephoneNumber:
 value +1 828 555 1212 already exists in entry uid=ajensen,ou=People,dc=example,dc=com
```

To Limit The Scope of Uniqueness

In some cases you need attribute uniqueness separately for different base DNs in your directory. For example, you need all `uid` values to remain unique both for users in `dc=example,dc=com` and `dc=example,dc=org`, but it is not a problem to have one entry under each base DN with the same user ID as the organizations are separate. The following steps demonstrate how to limit the scope of uniqueness by creating separate configuration entries for the unique attribute plugin.

1. If the attribute you target is not indexed for equality by default, index the attribute for equality.

   See "Configuring and Rebuilding Indexes" for instructions.

   The examples in this procedure target the user ID attribute, `uid`, which is indexed for equality by default.

2. For each base DN, set up a configuration entry that ensures the target attribute values are unique:

```
$ dsconfig \
 create-plugin \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "Unique Example.com UIDs" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --set type:uid \
 --trustAll \
 --no-prompt
$ dsconfig \
 create-plugin \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "Unique Example.org UIDs" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:dc=example,dc=org \
 --set type:uid \
 --trustAll \
 --no-prompt
```

3. Check that the plugin is working correctly:

```
$ cat unique-ids.ldif
dn: uid=unique,ou=People,dc=example,dc=com
uid: unique
givenName: Unique
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Unique Person
sn: Person
userPassword: 1Mun1qu3

dn: uid=unique,ou=People,dc=example,dc=org
uid: unique
givenName: Unique
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Unique Person
sn: Person
userPassword: 1Mun1qu3

dn: uid=copycat,ou=People,dc=example,dc=com
uid: unique
uid: copycat
```

```
givenName: Copycat
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Copycat Person
sn: Person
userPassword: copycopy

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 unique-ids.ldif
# Processing ADD request for uid=unique,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=unique,ou=People,dc=example,dc=com
# Processing ADD request for uid=unique,ou=People,dc=example,dc=org
# ADD operation successful for DN uid=unique,ou=People,dc=example,dc=org
# Processing ADD request for uid=copycat,ou=People,dc=example,dc=com
# ADD operation failed
# The LDAP modify request failed:  19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value unique
 already exists in entry uid=unique,ou=People,dc=example,dc=com
```

To Ensure Unique Attribute Values With Replication

The unique attribute plugin ensures unique attribute values on the directory server where the attribute value is updated. If client applications write the same attribute value separately at the same time on different directory replicas, it is possible that both servers consider the duplicate value unique, especially if the network is down between the replicas.

1. Enable the plugin identically on all replicas.

2. To avoid duplicate values where possible, try one of the following solutions:

    •   Use a load balancer or proxy technology to direct all updates to the unique attribute to the same directory server.

        The drawback here is the need for an additional component to direct the updates to the same server, and to manage failover should that server go down.

    •   Configure safe read mode assured replication between replicas storing the unique attribute.

        The drawbacks here are the cost of safe read assured replication, and the likelihood that assured replication can enter degraded mode during a network outage, thus continuing to allow updates during the outage.

Chapter 14
Managing Schema

This chapter describes how to manage Lightweight Directory Access Protocol (LDAP) schema definitions for directory data. In this chapter you will learn to:

• Understand LDAP schemas including the schema definitions delivered with OpenDJ servers

• Change and extend OpenDJ LDAP schemas

• Relax schema checking when troubleshooting data that does not conform to schema definitions

Schema definitions describe the data, and especially the object classes and attribute types that can be stored in the directory. By default, OpenDJ servers conform strictly to LDAPv3 standards pertaining to schema definitions and attribute syntax checking, ensuring that data stored is valid and properly formed. Unless your data uses only standard schema present in the server when you install, then you must add additional schema definitions to account for the data your applications stored.

OpenDJ servers come with many standard schema definitions out of the box. In addition, you can update and extend schema definitions while OpenDJ servers are online. As a result you can add new applications requiring additional data without stopping your directory service.

14.1. About Directory Schema

Directory schema, described in RFC 4512, defines the kinds of information you find in the directory, and can define how the information are related. This chapter focuses primarily on the following types of directory schema definitions:

• *Attribute type* definitions describe attributes of directory entries, such as `givenName` or `mail`.

  Here is an example of an attribute type definition:

```
# Attribute type definition
attributeTypes: ( 0.9.2342.19200300.100.1.3 NAME ( 'mail' 'rfc822Mailbox' )
  EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} X-ORIGIN 'RFC 4524' )
```

  Attribute type definitions start with an OID, and generally a short name or names that are easier to remember than the OID. The attribute type definition can specify how attribute values should be

collated for sorting, and what syntax they use. The X-ORIGIN is an extension to identify where the definition originated. When you define your own schema, you likely want to provide an X-ORIGIN to help you to track versions of definitions, and where the definitions came from.

Attribute type definitions indicate whether the attribute is a *user attribute* intended to be modified by external applications (the default, or specified with `USAGE userApplications`), or a *operational attribute* intended to be managed by the server for internal purposes (specified, for example, with `USAGE directoryOperation`). The user attributes that are required and permitted on each entry are defined by the entry's object classes. The server checks what the entry's object classes require and permit when it handles requests to update user attributes. No such check is performed for operational attributes.

Attribute type definitions differentiate operational attributes with the following `USAGE` types:

- `USAGE directoryOperation` indicates a generic operational attribute.

  This is most likely the type of operational attribute that you would define if necessary. This is the type used, for example, when creating a last login time attribute in "Track Last Login Time".

- `USAGE dSAOperation` indicates a DSA-specific operational attribute, meaning an operational attribute specific to the current server.

- `USAGE distributedOperation` indicates a DSA-shared operational attribute, meaning an operational attribute shared by multiple servers.

- *Object class* definitions identify the attribute types that an entry must have, and may have. Examples of object classes include `person` and `organizationalUnit`.

  Here is an example of an object class definition:

```
# Object class definition
objectClasses: ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL MUST ( sn $ cn )
  MAY ( userPassword $ telephoneNumber $ seeAlso $ description )
  X-ORIGIN 'RFC 4519' )
```

  Entries all have an attribute identifying their object classes, called `objectClass`.

  Object class definitions start with an object identifier (OID), and generally a short name that is easier to remember than the OID. The definition here says that the person object class inherits from the top object class, which is the top-level parent of all object classes. An entry's `objectclass` attribute lists the entry's object classes. An entry can have one STRUCTURAL object class inheritance branch, such as `top` - `person` - `organizationalPerson` - `inetOrgPerson`. Yet entries can have multiple AUXILIARY object classes. The object class then defines the attribute types that must be included, and the attribute types that may be included on entries having the object class.

- An *attribute syntax* constrains what directory clients can store as attribute values.

  An attribute syntax is identified in an attribute type definition by its OID. String-based syntax OIDs are optionally followed by a number set between braces that represents a minimum upper bound

on the number of characters in the attribute value. For example, in the attribute type definition shown above, the syntax is `1.3.6.1.4.1.1466.115.121.1.26{256}`. The syntax is an IA5 string (composed of characters from the international version of the ASCII character set) that can contain at least 256 characters.

You can find a table matching attribute syntax OIDs with their human-readable names in RFC 4517, Appendix A. Summary of Syntax Object Identifiers. The RFC describes attribute syntaxes in detail. Alternatively, you can see the attribute syntaxes that a server supports by opening the OpenDJ control panel and browsing to Schema > Manage Schema > Attribute Syntaxes. You can also list them by using the **dsconfig** command.

Although attribute syntaxes are often specified in attribute type definitions, directory servers do not always check that attribute values comply with attribute syntaxes. An OpenDJ server does tend to enforce compliance by default, in particular for certificates, country strings, directory strings, JPEG photos, and telephone numbers. The aim is to avoid accumulating garbage in your directory data.

If you are trying unsuccessfully to import non-compliant data from a more lenient directory server, you can either clean the data before importing it, or if cleaning the data is not an option, read "Relaxing Schema Checking to Import Legacy Data".

When creating your own attribute type definitions, use existing attribute syntaxes where possible. If you must create your own attribute syntax, then consider the extensions in Extensions for Attribute Syntax Descriptions.

• Matching rules determine how the directory server compares attribute values to assertion values for LDAP search and LDAP compare operations.

For example, suppose you search with the filter `(uid=bjensen)`. The assertion value in this case is `bjensen`.

OpenDJ has the following schema definition for the user ID attribute:

```
attributeTypes: ( 0.9.2342.19200300.100.1.1 NAME ( 'uid' 'userid' )
  EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} X-ORIGIN 'RFC 4519' )
```

When finding an equality match for your search, OpenDJ uses the `caseIgnoreMatch` matching rule to check for user ID attribute values that equal `bjensen` without regard to case.

You can see the matching rules that OpenDJ supports by opening the OpenDJ control panel and browsing to Schema > Manage Schema > Matching Rules. Notice that many matching rules support string collation in languages other than English. You can also list matching rules by using the **dsconfig** command.

As you can read in examples such as "Search: Listing Active Accounts" in the *Developer's Guide*, OpenDJ matching rules enable directory clients to compare other values besides strings, for example.

OpenDJ exposes schema over protocol through the `cn=schema` entry. OpenDJ stores the schema definitions corresponding to the entry in LDIF under the `config/schema/` directory. Many standard definitions and definitions pertaining to the server configuration are included at installation time.

14.2. Updating Directory Schema

OpenDJ servers are designed to permit updating the list of directory schema definitions while the server is running. As a result you can add support for new applications that require new attributes or new kinds of entries without interrupting the directory service. OpenDJ servers also replicate schema definitions, so the schema you add on one replica is propagated to other replicas without the need for manual intervention.

As it is easy to introduce typos into schema definitions, the best way to start defining your own schema is with the OpenDJ Control Panel. Open the control panel > Schema > Manage Schema window as shown in "Manage Schema Window" to get started creating custom object classes and attribute types.

Manage Schema Window
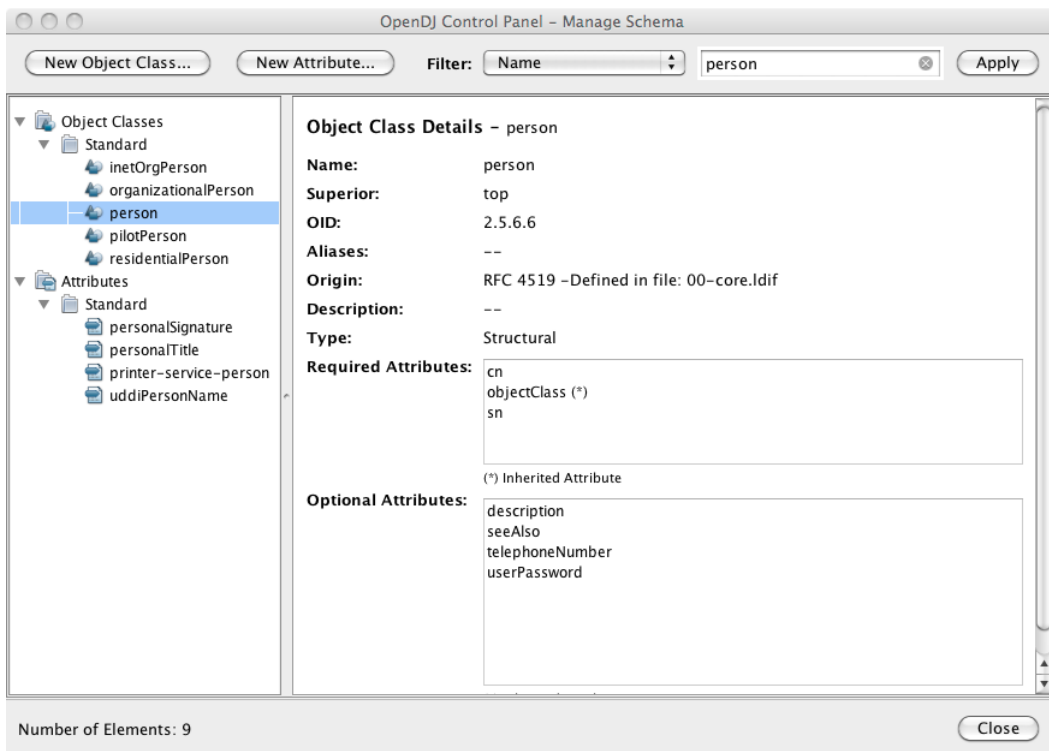
As object classes reference attribute types, you first create custom attribute types, and then create the object class that references the attribute types.

Create a custom attribute type through the New Attribute window shown in "New Attribute Window".

New Attribute Window

Using the New Object Class window shown in "New Object Class Window", create an auxiliary object class that allows your new custom attribute type. Set the type to Auxiliary under Extra Options.

New Object Class Window

When you finish, the schema changes show up by default in the file `config/schema/99-user.ldif`. Notice that the file name starts with a number, 99. This number is larger than the numbers prefixing other schema file names. In fact, OpenDJ servers read the schema files in sorted order, reading schema definitions as they occur. If a server reads a schema definition for an object class before it has read the definitions of the attribute types mentioned in the object class definition, then it displays an error. Therefore, when naming your schema file, make sure the name appears in the sorted list of file names *after* all the schema files containing definitions that your schema definitions depends on. The default file name for your schema, `99-user.ldif`, ensures that your definitions load only after all of the schema files installed by default.

You can create this file in the lab using the control panel, and then apply the definitions in production by adapting the content for use with the **ldapmodify** command, for example:

```
$ cd /path/to/opendj/config/schema
$ cat 99-user.ldif
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( temporary-fake-attr-id
  NAME 'myCustomAttribute'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications )
objectClasses: ( temporary-fake-oc-id
  NAME 'myCustomObjClass'
  SUP top
  AUXILIARY
  MAY myCustomAttribute )
```

To test your schema definition, add the object class and attribute to an entry:

```
$ cat custom-attr.ldif
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: myCustomObjClass
-
add: myCustomAttribute
myCustomAttribute: Testing 1, 2, 3...

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 custom-attr.ldif
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 myCustomAttribute
dn: uid=bjensen,ou=People,dc=example,dc=com
myCustomAttribute: Testing 1, 2, 3...
```

In addition to supporting the standard schema definitions that are described in RFC 4512, section 4.1, OpenDJ also supports the following extensions that you can use when adding your own definitions:

Extensions for All Schema Definitions

X-ORIGIN

Used to specify the origin of a schema element. Examples include `X-ORIGIN 'RFC 4519'`, `X-ORIGIN 'draft-ietf-ldup-subentry'`, and `X-ORIGIN 'OpenDJ Directory Server'`.

X-SCHEMA-FILE

Used to specify the relative path to the schema file containing the schema element such as `X-SCHEMA-FILE '00-core.ldif'`. Schema definitions are located by default in `/path/to/opendj/config/schema/*.ldif` files.

Extensions for Attribute Syntax Descriptions

X-ENUM

Used to define a syntax that is an enumeration of values. The following attribute syntax description defines a syntax allowing four possible attribute values, for example:

```
ldapSyntaxes: ( security-label-syntax-oid DESC 'Security Label'
 X-ENUM ( 'top-secret' 'secret' 'confidential' 'unclassified' ) )
```

X-PATTERN

Used to define a syntax based on a regular expression pattern, where valid regular expressions are those defined for `java.util.regex.Pattern`. The following attribute syntax description defines a simple, lenient SIP phone URI syntax check:

```
ldapSyntaxes: ( simple-sip-uri-syntax-oid DESC 'Lenient SIP URI Syntax'
 X-PATTERN '^sip:[a-zA-Z0-9.]+@[a-zA-Z0-9.]+(:[0-9]+)?$' )
```

X-SUBST

Used as a fallback to substitute a defined syntax for one that OpenDJ servers do not implement. The following example substitutes Directory String syntax, which has OID 1.3.6.1.4.1.1466.115.121.1.15, for a syntax that OpenDJ servers do not implement:

```
ldapSyntaxes: ( non-implemented-syntax-oid DESC 'Not Implemented in OpenDJ'
 X-SUBST '1.3.6.1.4.1.1466.115.121.1.15' )
```

Extension for Attribute Type Descriptions

X-APPROX

`X-APPROX` is used to specify the approximate matching rule to use for a given attribute type when not using the default, which is the double metaphone approximate match.

14.3. Relaxing Schema Checking to Import Legacy Data

By default, OpenDJ servers accept data that follows the schema for allowable and rejected data. You might have legacy data from a directory service that is more lenient, allowing non-standard constructions such as multiple structural object classes per entry, not checking attribute value syntax, or even not respecting schema definitions.

For example, when importing data with multiple structural object classes defined per entry, you can relax schema checking to warn rather than reject entries having this issue:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set single-structural-objectclass-behavior:warn \
 --trustAll \
 --no-prompt
```

You can allow attribute values that do not respect the defined syntax with the **dsconfig** command as well:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set invalid-attribute-syntax-behavior:warn \
 --trustAll \
 --no-prompt
```

You can even turn off schema checking altogether, although turning off schema checking only really makes sense when you are absolutely sure that the entries and attribute values respect the schema definitions, and you simply want to turn off schema checking temporarily to speed up import processing:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set check-schema:false \
 --trustAll \
 --no-prompt
```

14.4. Standard Schema Included With OpenDJ Server

OpenDJ servers provide many standard schema definitions in these LDIF files under `/path/to/opendj/config/schema`:

`00-core.ldif`

> This file contains a core set of attribute type and object class definitions from the following Internet-Drafts, RFCs, and standards:
>
> draft-boreham-numsubordinates
> draft-findlay-ldap-groupofentries
> draft-good-ldap-changelog
> draft-howard-namedobject
> draft-ietf-ldup-subentry
> draft-wahl-ldap-adminaddr
> RFC 1274
> RFC 2079
> RFC 2256
> RFC 2798
> RFC 3045
> RFC 3296
> RFC 3671
> RFC 3672
> RFC 4512
> RFC 4519
> RFC 4523
> RFC 4524
> RFC 4530
> RFC 5020
> X.501

`01-pwpolicy.ldif`

> This file contains schema definitions from draft-behera-ldap-password-policy (Draft 09), which defines a mechanism for storing password policy information in an LDAP directory server.

`02-config.ldif`

> This file contains the attribute type and objectclass definitions for use with the server configuration.

`03-changelog.ldif`

> This file contains schema definitions from draft-good-ldap-changelog, which defines a mechanism for storing information about changes to directory server data.

`03-rfc2713.ldif`

This file contains schema definitions from RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.

`03-rfc2714.ldif`

This file contains schema definitions from RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.

`03-rfc2739.ldif`

This file contains schema definitions from RFC 2739, which defines a mechanism for storing calendar and vCard objects in the directory server. Note that the definition in RFC 2739 contains a number of errors, and this schema file has been altered from the standard definition in order to fix a number of those problems.

`03-rfc2926.ldif`

This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.

`03-rfc3112.ldif`

This file contains schema definitions from RFC 3112, which defines the authentication password schema.

`03-rfc3712.ldif`

This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.

`03-uddiv3.ldif`

This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.

`04-rfc2307bis.ldif`

This file contains schema definitions from draft-howard-rfc2307bis, which defines a mechanism for storing naming service information in the directory server.

`05-rfc4876.ldif`

This file contains schema definitions from RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.

`05-samba.ldif`

This file contains schema definitions required when storing Samba user accounts in the directory server.

05-solaris.ldif

This file contains schema definitions required for Solaris and OpenSolaris LDAP naming services.

06-compat.ldif

This file contains the attribute type and objectclass definitions for use with the server configuration.

FORGEROCK

Chapter 15
Configuring LDAP Proxy Services

This chapter shows how to configure a server to forward LDAP requests to remote directory servers. For instructions on setting up a server as a proxy server, see "*Installing a Directory Proxy Server*" in the *Installation Guide* instead. In this chapter you will learn how to:

- Configure a discovery mechanism for remote directory servers

- Configure routing to remote directory servers

- Choose an appropriate load balancing algorithm

- Align schema definitions between a proxy and remote directory servers

- Configure a proxy backend that forwards LDAP requests

- Use access controls and resource limits with proxy services

- Use proxy services to deploy a highly available directory service

- Use proxy services to provide a single point of access to a directory service

15.1. About Proxy Services

ForgeRock Directory Services proxy services enable you to build a single point of access to a directory service, with a uniform view of the underlying LDAPv3 user data that hides implementation details from directory client applications, and promotes scalability and availability.

You can set up an OpenDJ server as a directory proxy server. For details, see "*Installing a Directory Proxy Server*" in the *Installation Guide*. When you set up a directory proxy server, no user data is stored locally. The server acts purely as an LDAP proxy. The only local data is for the server configuration and LDAP schema. In addition, the server is set up to use global access control policies rather than global ACIs. Global access control policies provide coarse-grained access control suitable for use on proxy servers, where the lack of local access to directory data makes ACIs a poor fit. For details, see "About Global Access Control Policies".

> **Note**
>
> Access control instructions (ACI) and global access control policies rely on different access control handlers, which implement different access control models. ACIs rely on the DSEE-compatible access control handler. (DSEE refers to Sun Java System Directory Server Enterprise Edition.) Global access control policies rely on the policy-based access control handler. A server can only use one handler at a time.

> Take the following constraints into consideration:
>
> • When the policy-based handler is configured, ACIs have no effect.
>
> • When the DSEE-compatible handler is configured, global access control policies have no effect.
>
> • When a server is set up as a directory server, it uses the DSEE-compatible access control handler, with ACIs in directory data and global ACIs in the configuration.
>
> • When a server is set up as a directory proxy server, it uses the policy-based access control policy handler, and global access control policies.
>
> • Once the server has been set up, the choice of access control handler cannot be changed with the **dsconfig** command.

The rest of this chapter describes how to configure proxy services when the server has already been set up.

Proxy services are provided by proxy backends. Proxy backends connect to remote directory servers using a dynamic and configurable discovery mechanism as described in "Discovering Remote Directory Servers". They route requests to remote directory servers as described in "Routing Requests to Remote Directory Servers". The way they distribute requests is configurable as described in "Choosing a Load Balancing Algorithm". The way they handle failures when processing forwarded requests is described in "Understanding How Failures Are Handled".

LDAP schema definitions for user data must be aligned on the proxy server and on the remote directory servers. For more information, see "Managing Schema Definitions".

ACIs are handled by the directory server where the target data is stored. In other words, global access control policies set on a proxy server do not change ACIs on the remote directory server. Set ACIs appropriately on the directory server independently of proxy settings.

You can set up a proxy backend as described in "Configuring a Proxy Backend". If you require only one proxy backend, set up the server as a directory proxy server. For details, see "*Installing a Directory Proxy Server*" in the *Installation Guide*. For additional deployment suggestions, see "Deploying Proxy Services for High Availability" and "Deploying a Single Point of Directory Access".

15.2. Discovering Remote Directory Servers

When the target DN of an LDAP request is not in a local backend, an LDAP server can refuse to handle the request, or return a referral. An LDAP proxy can also forward the request to another directory server.

In ForgeRock Directory Services, the LDAP proxy is implemented as a proxy backend. Rather than store user data locally, the proxy backend forwards requests to remote directory servers.

For proxy backends, a service discovery mechanism identifies remote directory servers to forward LDAP requests to. A service discovery mechanism's configuration specifies the keys used for secure

communications, and how to contact the remote directory servers. It reads remote directory servers' configurations to discover their capabilities, including the naming contexts they serve, and so which target DNs they can handle. It periodically rereads their configurations in case they have been updated since the last service discovery operation.

When preparing to configure a service discovery mechanism, choose one of these alternatives:

Replication service discovery mechanism

This mechanism contacts OpenDJ replication servers to discover directory servers to forward LDAP requests to. Each replication server maintains information about the replication topology that allows the proxy server to discover directory server replicas.

This mechanism only works with replicated OpenDJ servers.

A replication service discovery mechanism configuration includes a bind DN and password to connect to replication servers. It uses this account to read configuration data under `cn=admin data` and `cn=config`. The account must have access and privileges to read that configuration data, and it must exist with the same credentials on all replication servers.

Static service discovery mechanism

This mechanism maintains a static list of directory server *host:port* combinations. You must enumerate the servers to forward LDAP requests to.

This mechanism is designed to work with all LDAPv3 directory servers that support proxied authorization.

When configuring a service discovery mechanism, make sure that all the remote directory servers are replicas of each other, and that they have the same capabilities. A directory proxy server using this mechanism expects all remote directory servers to be equivalent.

The following example creates a replication service discovery mechanism that specifies two replication servers to contact securely:

```
$ dsconfig \
 create-service-discovery-mechanism \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --mechanism-name "Replication Service Discovery Mechanism" \
 --type replication \
 --set bind-dn:"uid=admin,cn=Administrators,cn=admin data" \
 --set bind-password:password \
 --set replication-server:rs1.example.com:4444 \
 --set replication-server:rs2.example.com:4444 \
 --set use-ssl:true \
 --set trust-manager-provider:"JVM Trust Manager" \
 --trustAll \
 --no-prompt
```

The following example creates a static service discovery mechanism that specifies four directory servers to contact:

```
$ dsconfig \
 create-service-discovery-mechanism \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --mechanism-name "Static Service Discovery Mechanism" \
 --type static \
 --set primary-server:local1.example.com:636 \
 --set primary-server:local2.example.com:636 \
 --set secondary-server:remote1.example.com:636 \
 --set secondary-server:remote2.example.com:636 \
 --set use-ssl:true \
 --set trust-manager-provider:"JVM Trust Manager" \
 --trustAll \
 --no-prompt
```

The examples above assume that the remote servers use certificates signed by well-known CAs, and so are recognized using the trust manager provided by the JVM. If this is not the case, configure an appropriate trust manager provider.

If the proxy server must perform SSL mutual authentication when setting up secure connections with the remote servers, configure an appropriate key manager provider and SSL certificate nickname.

Supported service discovery mechanisms define a `discovery-interval` that specifies how often to read the remote server configurations in order to discover changes. Because the mechanism polls periodically for configuration changes, by default, it can take up to one minute for the mechanism to see the changes. If necessary, you can change this setting in the configuration.

15.3. Routing Requests to Remote Directory Servers

A proxy backend forwards requests according to their target DNs. The proxy backend matches target DNs to base DNs that you specify in the configuration.

When specifying base DNs, bear in mind the following points:

- A server responds first with local data, such as the server's own configuration or monitoring data. If a request target DN cannot be served from local data, the proxy backend can forward the request to a remote directory server.

  The proxy backend will forward the request if its target DN is under one of the specified base DNs.

- As an alternative to enumerating a list of base DNs to proxy, you can set the proxy backend property `route-all: true`.

  When you activate this property, the proxy backend will attempt to forward all requests that can be served by public naming contexts of remote servers. If the request target DN is not in a naming

context supported by any remote directory servers associated with the proxy backend, then the proxy will not forward the request.

- The LDAP proxy capability is intended to proxy user data, not server-specific data.

  A server with a proxy backend still responds directly to requests that target private naming contexts such as `cn=config`, `cn=tasks`, and `cn=monitor`. Local backends hold configuration and monitoring information that is specific to the server, and these naming contexts are not public.

  Make sure that client applications for configuration and monitoring access servers directly for the server-specific data they need.

- If you configure multiple proxy backends, each proxy backend must target distinct base DNs.

15.4. Choosing a Load Balancing Algorithm

A directory proxy server balances the request load across the remote directory servers it forwards requests to. The load balancing algorithm is part of the proxy backend configuration.

When configuring a proxy backend, choose one of these load balancing alternatives:

`affinity`

This load balancing algorithm routes requests with the same target DN to the same server.

Affinity load balancing helps when applications update and then reread the same entry in quick succession. With an add or modify request on an entry that is quickly followed by a read of the entry, the request to replicate the update can take longer than the read request, depending on network latency. Affinity load balancing forwards the read request to the same server that processed the update, ensuring that the client obtains the expected result.

In terms of the CAP theorem, affinity load balancing provides consistency and availability, but not partition tolerance. As this algorithm lacks partition tolerance, only use it to load balance requests in environments where partitions are unlikely, such as a single data center with all remote directory servers on the same network.

`least-requests`

This load balancing algorithm routes requests to the LDAP server with the fewest active requests from the current directory proxy server.

Least requests load balancing helps to spread requests equitably across a pool of replicated servers.

In terms of the CAP theorem, least requests load balancing provides availability and partition tolerance, but not consistency. A write request followed by a read request of the same entry might be forwarded to different remote directory servers.

If a remote directory server becomes unavailable, the proxy server can fail over to use another remote directory server. This prevents the proxy server from continuing to forward requests to a remote directory server that cannot handle them. In addition, the available service discovery mechanisms let you specify some remote directory servers as primary (preferred) servers, and others as secondary servers. This makes it possible to fail over first to servers in the same data center, and only if necessary to servers in a remote data center. For more information, see "Understanding How Failures Are Handled".

15.5. Managing Schema Definitions

Proxy services are designed to proxy user data rather than server configuration or monitoring data. In addition, a proxy server must expose the same LDAP schema for user data as the remote directory servers.

If the schema definitions for user data differ between the proxy server and the remote directory servers, you must update them to bring them into alignment.

For details on OpenDJ server schema, see "*Managing Schema*".

If the remote servers are not OpenDJ servers, also see the schema documentation for the remote servers. Schema formats are not identical across all LDAPv3 servers. You will need to translate the differences into native formats, and apply changes locally on each server.

15.6. Configuring a Proxy Backend

This section shows how to create a proxy backend once you have made your configuration choices. As described in "About Proxy Services", how the server was set up determines the access control model. The access control model for proxy servers is mutually exclusive with the model for directory servers.

Create proxy backends only on servers set up as proxy servers (with **setup proxy-server**).

> **Important**
>
> A directory proxy server uses a proxy user DN and password to connect to remote directory servers, and proxied authorization for forwarded LDAP requests. This proxy account must exist with the same credentials on all remote directory servers, and must be able to use the standard proxied authorization control.
>
> For details on proxied authorization and how to configure OpenDJ directory servers to allow it, see "Configuring Proxied Authorization" in the *Developer's Guide*.

This section includes the following examples:

- "Proxy: Forward Requests for Example.com"

- "Proxy: Forward All User Data Requests"

Proxy: Forward Requests for Example.com

The following example creates a proxy backend that forwards LDAP requests when the target DN is in dc=example,dc=com. It uses the replication service discovery mechanism described in "Discovering Remote Directory Servers", and affinity load balancing described in "Choosing a Load Balancing Algorithm":

```
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name proxyExampleCom \
 --type proxy \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --set route-all:false \
 --set proxy-user-dn:cn=proxy,ou=apps,dc=example,dc=com \
 --set proxy-user-password:password \
 --set load-balancing-algorithm:affinity \
 --set service-discovery-mechanism:"Replication Service Discovery Mechanism" \
 --trustAll \
 --no-prompt
```

This command fails if dc=example,dc=com is already served by local data.

Proxy: Forward All User Data Requests

The following example creates a proxy backend that forwards LDAP requests targeting user data, not specifying base DNs explicity, but instead using the route-all property. It uses the static service discovery mechanism described in "Discovering Remote Directory Servers", and least requests load balancing described in "Choosing a Load Balancing Algorithm":

```
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name proxyAll \
 --type proxy \
 --set enabled:true \
 --set route-all:true \
 --set proxy-user-dn:cn=proxy,ou=apps,dc=example,dc=com \
 --set proxy-user-password:password \
 --set load-balancing-algorithm:least-requests \
 --set service-discovery-mechanism:"Static Service Discovery Mechanism" \
 --trustAll \
 --no-prompt
```

The examples above do not specify how to secure proxy connections to remote directory servers. Connection security is defined using configuration properties of the service discovery mechanism that the proxy backend depends on.

Proxy backends define a `discovery-interval` that specifies how often to read the remote server configurations in order to discover changes. Because the proxy polls periodically for configuration changes, by default, it can take up to one minute to see the changes. If necessary, you can change this setting in the configuration.

15.7. Understanding How Failures Are Handled

A proxy backend protects against failed and stale connections with feedback from periodic heartbeat requests to remote directory servers.

A heartbeat request serves these purposes:

• Checks that the remote directory server is still available.

   If the request fails, the proxy closes the unresponsive connection and connects to another remote directory server.

• Determines whether an unresponsive server has recovered.

   When the remote directory server responds again to heartbeat requests, the proxy can begin forwarding client requests to it again.

• Keeps the connection alive, preventing it from appearing idle and being forcefully closed.

If necessary, you can configure how often heartbeat requests are sent using the proxy backend `heartbeat-interval` property.

15.8. Applying Access Control and Resource Limits for Proxy Services

When you deploy a pure directory proxy server, where the server has only proxy backends and configuration and no local user data, you limit the mechanisms you can use to control access and limit resource use. Such mechanisms cannot rely on ACIs in user data, resource limit settings on user entries, or group membership to determine what the server should allow. They can depend only on the server configuration and on the properties of the incoming request.

Global access control policies provide an access control mechanism that is suitable for pure directory proxy servers. For details, see "About Global Access Control Policies".

Resource limits set in the server configuration provide a way to prevent directory clients from using an unfair share of system resources. For details, see "*Setting Resource Limits*", which covers different methods, including how to update a server's configuration.

15.9. Deploying Proxy Services for High Availability

Directory services are designed for basic availability. Directory data replication makes it possible to continue reading and writing directory data even when the network is partitioned, meaning that remote servers cannot effectively contact each other. This sort of availability assumes, however, that client applications can handle temporary interruptions.

Some client applications can be configured to connect to a set of directory servers. Some of these clients are able to retry requests when a server is unavailable. Others expect a single point of entry to the directory service, or cannot continue promptly when a directory server is unavailable.

Individual directory servers can become unavailable for various reasons:

• A network problem can make it impossible to reach the server.

• The server can be temporarily down for maintenance (backup, upgrade, and other purposes).

• The server can be removed from a replication topology and replaced with a different server.

• A crash can bring the server down temporarily for a restart or permanently for a replacement.

All of these interruptions must be managed on the client side. Some could require reconfiguration of each client application.

A directory proxy server provides high availability to client applications by hiding these implementation details from client applications. The proxy presents client applications with a uniform view of the remote directory servers. It periodically rereads the remote servers' configurations so that it can route requests correctly even when remote servers change. It regularly checks connections to remote servers so that it can route requests correctly even when some servers are unavailable.

In addition, directory proxy servers are well-suited for applications that need a single entry point to the directory service. For details, see "Deploying a Single Point of Directory Access".

15.10. Deploying a Single Point of Directory Access

Unlike directory servers with their potentially large sets of volatile user data, standalone directory proxy servers manage only their configuration state. Proxy servers can start faster and require less disk and memory space. Each directory proxy server can effectively be a clone of every other, with a configuration that does not change after server startup. Each clone routes LDAP requests and handles problems in the same way.

When you configure all directory proxy servers as clones of each other, you have a number of identical directory access points. Each point provides the same view of the underlying directory service. The points differ from each other only by their connection coordinates (host, port, and potentially key material to establish secure connections).

To consolidate identical access points to a single access point, configure your network to use a virtual IP address for the proxy servers. You can restart or replace proxy servers at any time, in the worst case losing only the client connections that were established with the individual proxy server.

An alternative to a single, global access point for all applications is to repeat the same approach for each key application. The proxy server configuration is specific to each key application. Clones with that configuration provide a single directory access point for the key application. Other clones do the same other for other key applications. Be sure to provide a more generic access point for additional applications.

Chapter 16

Configuring Pass-Through Authentication

This chapter focuses on pass-through authentication (PTA), whereby you configure another server to determine the response to an authentication request. A typical use case for PTA involves passing authentication through to Active Directory for users coming from Microsoft Windows systems. In this chapter you will learn to:

• Configure password policies to use PTA

• Assign PTA policies to users and to groups

16.1. About Pass-Through Authentication

You use *pass-through authentication* (PTA) when the credentials for authenticating are stored in a remote directory service instead of an OpenDJ service. In effect, the OpenDJ server redirects the bind operation against a remote LDAP server.

The method an OpenDJ server uses to redirect the bind depends on the mapping from the user entry in the OpenDJ server to the corresponding user entry in the remote directory. OpenDJ servers provide you several choices to set up the mapping:

• When both the local entry in the OpenDJ server and the remote entry in the other server have the same DN, you do not have to set up the mapping. By default, the OpenDJ server redirects the bind with the original DN and password from the client application.

• When the local entry in the OpenDJ server has been provisioned with an attribute holding the DN of the remote entry, you can specify which attribute holds the DN, and the OpenDJ server redirects the bind on the remote server using the DN value.

• When you cannot get the remote bind DN directly, you need an attribute and value on the OpenDJ entry that corresponds to an identical attribute and value on the remote server. In this case, you also need the bind credentials for a user who can search for the entry on the remote server. The OpenDJ server performs a search for the entry using the matching attribute and value, and then redirects the bind with the DN from the remote entry.

You configure PTA as an authentication policy that you associate with a user's entry in the same way that you associate a password policy with a user's entry. Either a user has an authentication policy for PTA, or the user has a local password policy.

16.2. Setting Up Pass-Through Authentication

When setting up pass-through authentication, you need to know to which remote server or servers to redirect binds, and you need to know how you map user entries in the OpenDJ server to user entries in the remote directory.

To Set Up SSL Communication For Testing

When performing PTA, you protect communications between the OpenDJ server and the server providing authentication. If you test secure connections with self-signed certificates, and you do not want the client to blindly trust the server, follow these steps to import the authentication server's certificate into the truststore used by the OpenDJ server.

1. Export the server certificate from the authentication server.

   How you perform this step depends on the authentication directory server. With an OpenDJ server, you can export the certificate as shown here:

   ```
   $ keytool \
    -exportcert \
    -rfc \
    -alias server-cert \
    -keystore /path/to/authn-server/config/keystore \
    -storepass:file /path/to/authn-server/config/keystore.pin \
    -storetype PKCS12 \
    -file authn-server-cert.pem
   ```

2. Record the host name used in the certificate.

   You use the host name when configuring the secure connection. With an OpenDJ server, you can view the certificate details as shown here:

   ```
   $ keytool \
    -list \
    -v \
    -alias server-cert \
    -keystore /path/to/authn-server/config/keystore \
    -storepass:file /path/to/authn-server/config/keystore.pin \
    -storetype PKCS12
   ...
   Owner: CN=authn-server.example.com, O=OpenDJ RSA Self-Signed Certificate
   Issuer: CN=authn-server.example.com, O=OpenDJ RSA Self-Signed Certificate
   ...
   ```

3. Import the authentication server certificate into OpenDJ's keystore:

```
$ keytool \
 -importcert \
 -alias authn-server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -file authn-server-cert.pem \
 -noprompt
```

To Configure an LDAP Pass-Through Authentication Policy

You configure authentication policies with the **dsconfig** command. Notice that authentication policies are part of the server configuration, and therefore not replicated.

1. Set up an authentication policy for pass-through authentication to the authentication server:

```
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "PTA Policy" \
 --type ldap-pass-through \
 --set primary-remote-ldap-server:authn-server.example.com:1636 \
 --set mapped-attribute:uid \
 --set mapped-search-base-dn:"dc=example,dc=com" \
 --set mapping-policy:mapped-search \
 --set use-ssl:true \
 --trustAll \
 --no-prompt
```

The policy shown here maps identities with this password policy to identities under `dc=example,dc=com` on the authentication server. Users must have the same `uid` values on both servers. The policy here also uses LDAPS between the OpenDJ server and the authentication server.

2. Check that your policy has been added to the list:

```
$ dsconfig \
 list-password-policies \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --property use-ssl \
 --trustAll \
 --no-prompt
Password Policy         : Type            : use-
ssl
------------------------:-----------------:--------
Default Password Policy : password-policy   : -
PTA Policy              : ldap-pass-through : true
Root Password Policy    : password-policy   : -
```

To Configure Pass-Through Authentication To Active Directory

The steps below demonstrate how to set up PTA to Active Directory. Here is some information to help you make sense of the steps.

Entries on the OpenDJ side use `uid` as the naming attribute, and entries also have `cn` attributes. Active Directory entries use `cn` as the naming attribute. User entries on both sides share the same `cn` values. The mapping between entries therefore uses `cn`.

Consider the example where an OpenDJ account with `cn=LDAP PTA User` and DN `uid=ldapptauser,ou=People,dc=example,dc=com` corresponds to an Active Directory account with DN `CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com`. The steps below enable the user with `cn=LDAP PTA User` on the OpenDJ server to authenticate through Active Directory:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --baseDN dc=example,dc=com \
 uid=ldapptauser \
 cn
dn: uid=ldapptauser,ou=People,dc=example,dc=com
cn: LDAP PTA User

$ ldapsearch \
 --hostname ad.example.com \
 --baseDN "CN=Users,DC=internal,DC=forgerock,DC=com" \
 --bindDN "cn=administrator,cn=Users,DC=internal,DC=forgerock,DC=com" \
 --bindPassword password \
 "(cn=LDAP PTA User)" \
 cn
dn: CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com
cn: LDAP PTA User
```

The OpenDJ server must map its `uid=ldapptauser,ou=People,dc=example,dc=com` entry to the Active Directory entry, `CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com`. In order to do the mapping,

the OpenDJ server has to perform a search for the user in Active Directory using the `cn` value that it recovers from its own entry for the user. Active Directory does not allow anonymous searches, so part of the authentication policy configuration consists of the administrator DN and password the OpenDJ server uses to bind to Active Directory to search.

Finally, before setting up the PTA policy, make sure the OpenDJ server can connect to Active Directory over a secure connection to avoid sending passwords in the clear.

1. Export the certificate from the Windows server.

   a. Click start > All Programs > Administrative Tools > Certification Authority, then right-click the CA and select Properties.

   b. In the General tab, select the certificate and click View Certificate.

   c. In the Certificate dialog, click the Details tab, then click Copy to File...

   d. Use the Certificate Export Wizard to export the certificate into a file, such as `windows.cer`.

2. Copy the exported certificate to the system running the OpenDJ server.

3. Import the server certificate into the OpenDJ keystore:

```
$ cd /path/to/opendj/config
$ keytool \
 -importcert \
 -alias ad-cert \
 -keystore keystore \
 -storepass:file keystore.pin \
 -storetype PKCS12 \
 -file ~/Downloads/windows.cer \
 -noprompt
Certificate was added to keystore
```

   At this point, the OpenDJ server can connect securely to Active Directory.

4. Set up an authentication policy for OpenDJ users to authenticate to Active Directory:

```
# Create a trust manager provider to access the Active Directory certificate:
$ dsconfig \
 create-trust-manager-provider \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name PKCS12 \
 --type file-based \
 --set enabled:true \
 --set trust-store-type:PKCS12 \
 --set trust-store-file:config/keystore \
 --set trust-store-pin-file:config/keystore.pin \
 --trustAll \
```

```
--no-prompt

# Use the trust manager provider in the pass-through authentication policy:
$ dsconfig \
 create-password-policy \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --type ldap-pass-through \
 --policy-name "AD PTA Policy" \
 --set primary-remote-ldap-server:ad.example.com:636 \
 --set mapped-attribute:cn \
 --set mapped-search-base-dn:"CN=Users,DC=internal,DC=forgerock,DC=com" \
 --set mapped-search-bind-dn:"cn=administrator,cn=Users,DC=internal, \
  DC=forgerock,DC=com" \
 --set mapped-search-bind-password:password \
 --set mapping-policy:mapped-search \
 --set trust-manager-provider:PKCS12 \
 --set use-ssl:true \
 --trustAll \
 --no-prompt
```

5. Assign the authentication policy to a test user:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: uid=ldapptauser,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=AD PTA Policy,cn=Password Policies,cn=config

Processing MODIFY request for uid=ldapptauser,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=ldapptauser,ou=People,dc=example,dc=com
```

6. Check that the user can bind using PTA to Active Directory:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN uid=ldapptauser,ou=People,dc=example,dc=com \
 --bindPassword password \
 "(cn=LDAP PTA User)" \
 userpassword cn
dn: uid=ldapptauser,ou=People,dc=example,dc=com
cn: LDAP PTA User
```

Notice that to complete the search, the user authenticated with a password to Active Directory,
though no `userpassword` value is present on the entry on the OpenDJ side.

16.3. Assigning Pass-Through Authentication Policies

You assign authentication policies in the same way as you assign password policies, by using the `ds-pwp-password-policy-dn` attribute.

> **Note**
>
> Although you assign the pass-through authentication policy using the same attribute as for password policy, the authentication policy is not in fact a password policy. Therefore, the user with a pass-through authentication policy does not have a value for the operational attribute `pwdPolicySubentry`.

To Assign a Pass-Through Authentication Policy To a User

Users depending on PTA no longer need a local password policy, as they no longer authenticate locally.

Examples in the following procedure work for this user, whose entry in an OpenDJ directory server is as shown below. Notice that the user has no `userPassword` attribute. The user's password on the authentication server is `password`:

```
dn: uid=ptaUser,ou=People,dc=example,dc=com
uid: ptaUser
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetorgperson
cn: PTA User
givenName: PTA
sn: User
homePhone: +1 225 216 5900
mail: ptaUser@example.com
```

This user's entry on the authentication server also has `uid=ptaUser`. The pass-through authentication policy performs the mapping to find the user entry in the authentication server.

1. Give an administrator access to update a user's password policy:

   ```
   $ cat protect-pta.ldif
   dn: ou=People,dc=example,dc=com
   changetype: modify
   add: aci
   aci: (targetattr = "ds-pwp-password-policy-dn")
    (version 3.0;acl "Allow Kirsten Vaughan to assign password policies";
    allow (all) (userdn = "ldap:///uid=kvaughan,ou=People,dc=example,dc=com");)

   $ ldapmodify \
    --port 1389 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    protect-pta.ldif
   ```

2. Update the user's `ds-pwp-password-policy-dn` attribute:

```
$ cat pta-policy.ldif
dn: uid=ptaUser,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=PTA Policy,cn=Password Policies,cn=config

$ ldapmodify \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 pta-policy.ldif
```

3. Check that the user can authenticate through to the authentication server:

```
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN uid=ptaUser,ou=People,dc=example,dc=com \
 --bindPassword password \
 "(uid=ptaUser)" \
 1.1
dn: uid=ptaUser,ou=People,dc=example,dc=com
```

To Assign a Pass-Through Authentication Policy To a Group

Examples in the following steps use the PTA policy as defined above. The administrator's entry is present on the authentication server as well.

1. Give an administrator the privilege to write subentries, such as those used for password policies:

```
$ cat subentry-write.ldif
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 subentry-write.ldif
```

Notice here that the root DN user `cn=Directory Manager` assigns privileges to Kirsten Vaughan. Any administrator with the `privilege-change` privilege can assign other privileges. Assuming the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, avoid assigning the `privilege-change` privilege to normal administrator users.

2. Create a subentry for a collective attribute that sets the password policy `ds-pwp-password-policy-dn` attribute for group members' entries:

```
$ cat collective-pta.ldif
dn: cn=PTA Policy for Dir Admins,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: PTA Policy for Dir Admins
ds-pwp-password-policy-dn;collective: cn=PTA Policy,cn=Password Policies,cn=config
subtreeSpecification: { base "ou=People", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)"}

$ ldapmodify \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 collective-pta.ldif
```

3. Check that the OpenDJ server has applied the policy.

   a. Make sure you can bind as the user on the authentication server:

```
$ ldapsearch \
 --hostname authn-server.example.com \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=People,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "dc=example,dc=com" \
 "(uid=kvaughan)" \
 1.1
dn: uid=kvaughan,ou=People,dc=example,dc=com
```

   b. Check that the user can authenticate through to the authentication server from the OpenDJ server:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 1.1
dn: uid=kvaughan,ou=People,dc=example,dc=com
```

Chapter 17
Samba Password Synchronization

This chapter covers synchronization between directory passwords and Samba passwords. In this chapter you will learn to:

- Configure Samba for use with an OpenDJ server

- Set up the OpenDJ Samba password plugin for synchronization

Samba, the Windows interoperability suite for Linux and UNIX, stores accounts because UNIX and Windows password storage management is not interoperable. The default account storage mechanism is designed to work well with relatively small numbers of accounts and configurations with one domain controller. For larger installations, you can configure Samba to use an OpenDJ server for storing Samba accounts. See the Samba documentation for your platform for instructions on how to configure an LDAP directory server such as an OpenDJ server as a Samba `passdb` backend.

The rest of this chapter focuses on how you keep passwords in sync when using an OpenDJ server for Samba account storage.

When you store Samba accounts in an OpenDJ server, Samba stores its own attributes as defined in the Samba schema. Samba does not use the LDAP standard `userPassword` attribute to store users' Samba passwords. You can configure Samba to apply changes to Samba passwords to LDAP passwords as well, too. Yet, if a user modifies their LDAP password directly without updating the Samba password, the LDAP and Samba passwords get out of sync.

The OpenDJ Samba Password plugin resolves this problem for you. The plugin intercepts password changes to Samba user profiles, synchronizing Samba password and LDAP password values. For an incoming Password Modify Extended Request or modify request changing the user password, the OpenDJ Samba Password plugin detects whether the user's entry reflects a Samba user profile (entry has object class `sambaSAMAccount`), hashes the incoming password value, and applies the password change to the appropriate password attribute, keeping the password values in sync. The OpenDJ Samba Password plugin can perform synchronization as long as new passwords values are provided in cleartext in the modification request. If you configure Samba to synchronize LDAP passwords when it changes Samba passwords, then the plugin can ignore changes by the Samba user to avoid duplicate synchronization.

To Set Up a Samba Administrator Account

The Samba Administrator synchronizes LDAP passwords after changing Samba passwords by issuing a Password Modify Extended Request. In Samba's `smb.conf` configuration file, the value of `ldap admin`

`dn` is set to the DN of this account. When the Samba Administrator changes a user password, the plugin ignores the changes, so choose a distinct account different from Directory Manager and other administrators.

1. Create or choose an account for the Samba Administrator:

```
$ cat samba.ldif
dn: uid=Samba Admin,ou=Special Users,dc=example,dc=com
cn: Samba Administrator
givenName: Samba
mail: samba@example.com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
sn: Administrator
uid: Samba Admin
userPassword: password

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 samba.ldif
```

2. Ensure the Samba Administrator can reset user passwords:

```
$ cat samba-rights.ldif
dn: uid=Samba Admin,ou=Special Users,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")(targetattr ="*")
 (version 3.0; acl "Samba Admin user rights"; allow(all)
  userdn="ldap:///uid=Samba Admin,ou=Special Users,dc=example,dc=com";)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 samba-rights.ldif
```

To Set Up the Samba Password Plugin

1. Determine whether the plugin must store passwords hashed like LanManager (`sync-lm-password`) or like Windows NT (`sync-nt-password`), based on how you set up Samba in your environment.

2. Enable the plugin:

```
$ dsconfig \
 create-plugin \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "Samba Password Synchronisation" \
 --type samba-password \
 --set enabled:true \
 --set pwd-sync-policy:sync-nt-password \
 --set samba-administrator-dn:"uid=Samba Admin,ou=Special Users,dc=example,dc=com" \
 --trustAll \
 --no-prompt
```

At this point the Samba Password plugin is active.

3. (Optional) When troubleshooting Samba Password plugin issues, you can turn on debug logging as follows:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:true \
 --trustAll \
 --no-prompt
$ dsconfig \
 create-debug-target \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --target-name org.opends.server.plugins.SambaPasswordPlugin \
 --set enabled:true \
 --trustAll \
 --no-prompt
$ tail -f /path/to/opendj/logs/debug
```

Chapter 18
Monitoring, Logging, and Alerts

This chapter covers monitoring capabilities. In this chapter you will learn to:

• Access monitoring information remotely using LDAP, SNMP, or JMX.

• Monitor server status, including task status.

• Configure logs and interpret the messages they contain.

• Configure email settings for administrative alert notifications.

OpenDJ control panel provides basic monitoring capabilities under Monitoring > General Information, Monitoring > Connection Handler, and Monitoring > Manage Tasks. This chapter covers the other options for monitoring OpenDJ.

18.1. LDAP-Based Monitoring

OpenDJ servers publish monitoring information over LDAP under the entry `cn=monitor`. Many different types of information are exposed.

Interface stability: Evolving (See "ForgeRock Product Interface Stability" in the *Reference*)

The following example shows monitoring information for the `userRoot` backend holding Example.com data:

```
$ ldapsearch --port 1389 --baseDN cn=monitor "(cn=userRoot backend)"
dn: cn=userRoot backend,cn=Disk Space Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
disk-dir: /path/to/opendj/db/userRoot
disk-free: <bytes>
disk-state: normal
cn: userRoot backend

dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: false
ds-backend-entry-count: <count>
ds-base-dn-entry-count: <count> dc=example,dc=com
ds-backend-writability-mode: enabled
cn: userRoot Backend
```

In production systems, set a global ACI or access policy to grant access to `cn=monitor`, rather than allowing anonymous access.

18.2. SNMP-Based Monitoring

OpenDJ servers support SNMP, including the Management Information Base described in *RFC 2605: Directory Server Monitoring MIB*.

SNMP is not enabled by default. SNMP-based monitoring depends on OpenDMK, which you must download and install separately. This requires that you accept the Binary License for Project OpenDMK.

To obtain the OpenDMK self-extracting .jar file, download and unpack the ForgeRock Directory Services 5.5 .zip distribution from ForgeRock BackStage.

To run the OpenDMK installer, use the self-extracting .jar file that is part of the unpacked download. The following example unpacks the ForgeRock Directory Services 5.5 files into the default `opendj` directory, and then starts the OpenDMK installer:

```
$ cd ~/Downloads/
$ unzip DS-5.5.0.zip
$ java -jar opendj/snmp/opendmk.jar
```

If you install under `/path/to`, then the runtime library needed for SNMP is `/path/to/OpenDMK-bin/lib/jdmkrt.jar`.

After installing OpenDMK, set up an SNMP connection handler that uses your installation of the OpenDMK `jdmkrt.jar` library:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SNMP Connection Handler" \
 --set enabled:true \
 --set opendmk-jarfile:/path/to/OpenDMK-bin/lib/jdmkrt.jar \
 --trustAll \
 --no-prompt
```

By default, the SNMP connection handler listens on port 161 and uses port 162 for traps. On UNIX and Linux systems, only root can normally open these ports. To install as a normal user, change the listen and trap ports:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SNMP Connection Handler" \
 --set listen-port:11161 \
 --set trap-port:11162 \
 --trustAll \
 --no-prompt
```

Restart the SNMP connection handler to take the changes into account:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SNMP Connection Handler" \
 --set enabled:false \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SNMP Connection Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

To check that the response on the SNMP listen port, use a command such as **snmpwalk**:

```
$ snmpwalk -v 2c -c OpenDJ@OpenDJ localhost:11161
SNMPv2-SMI::mib-2.66.1.1.1.1 = STRING: "OpenDJ 4.0.0..."
SNMPv2-SMI::mib-2.66.1.1.2.1 = STRING: "/path/to/opendj"
...
```

18.3. JMX-Based Monitoring

OpenDJ servers support JMX-based monitoring. A number of tools support JMX, including the **jconsole** and **jvisualvm** commands bundled with the Sun/Oracle Java platform. JMX is not configured by default. Use the **dsconfig** command to configure the JMX connection handler:

Interface stability: Evolving (See "ForgeRock Product Interface Stability" in the *Reference*)

```
$ dsconfig \
 create-connection-handler \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "JMX Connection Handler" \
 --type jmx \
 --set enabled:true \
 --set listen-port:1689 \
 --trustAll \
 --no-prompt
```

By default, no users have privileges to access the JMX connection. The following command adds JMX privileges for Directory Manager:

```
$ dsconfig \
 set-root-dn-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --add default-root-privilege-name:jmx-notify \
 --add default-root-privilege-name:jmx-read \
 --add default-root-privilege-name:jmx-write \
 --trustAll \
 --no-prompt
```

Also configure security to login remotely. See the section on *Using SSL* in *Monitoring and Management Using JMX* for hints.

Alternatively, connect to a local server process using the process ID:

```
$ jvisualvm --openpid $(</path/to/opendj/logs/server.pid)
```

18.4. Server Operation and Tasks

OpenDJ servers have commands for monitoring server processes and tasks. The **status** command, described in status(1) in the *Reference*, displays basic information about the local server, similar to what is seen in the default window of the control panel. The **manage-tasks** command, described in manage-tasks(1) in the *Reference*, lets you manage tasks scheduled on a server, such as nightly backup.

The **status** command takes administrative credentials to read the configuration, as does the control panel:

```
$ status \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --script-friendly \
 --trustAll
Server Run Status: Started
Open Connections: 1
Host Name: opendj.example.com
Administrative Users: cn=Directory Manager
Installation Path: /path/to/opendj
Version: OpenDJ Server 4.0.0
Java Version: <version>
Administration Connector: Port 4444 (LDAPS)
-
Address:Port: --
Protocol: LDIF
State: Disabled
```

```
-
Address:Port: 0.0.0.0:161
Protocol: SNMP
State: Disabled
-
Address:Port: 0.0.0.0:8080
Protocol: HTTP
State: Disabled
-
Address:Port: 0.0.0.0:1389
Protocol: LDAP
State: Enabled
-
Address:Port: 0.0.0.0:1636
Protocol: LDAPS
State: Enabled
-
Address:Port: 0.0.0.0:1689
Protocol: JMX
State: Enabled
-
Base DN:      dc=example,dc=com
Backend ID:  userRoot
Entries:      <count>
Replication:
```

The **manage-tasks** command connects to the administration port, and so can connect to both local and remote servers:

```
$ manage-tasks \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt
```

18.5. Server Logs

By default, the server stores the following files under the `logs/` directory:

- Access logs for messages about clients accessing the server.

  One form of access log is a directory audit log. An audit log records changes to directory data in LDIF.

- Debug logs for messages tracing internal server events.

- Error logs for messages tracing server events.

- Replication logs for messages used to help repair problems in data replication.

- A `server.out` log for messages about server events since startup.

  Messages in this file have the same format as error log messages.

- A `server.pid` process ID file when the server is running.

Logs are handled by *log publishers*. Log publishers determine which messages to publish, where to publish them, and what output format to use.

The server's logging system supports extensibility through the ForgeRock Common Audit event framework. (Common Audit deals with any event you can audit, not only the directory data changes recorded in a directory audit log.) The ForgeRock Common Audit event framework provides log handlers for publishing to local files or to remote systems, as described in "Common ForgeRock Access Logs".

You configure logging using the **dsconfig** command. In addition to configuring how messages are published, you can configure policies for log file rotation and retention.

18.5.1. Access Logs

An *access log* traces client requests that the server processes. Each message includes a datestamp, information about the connection, and information about the operation processed.

> **Tip**
>
> It is possible to configure multiple access logs at the same time. This makes it possible to have a primary unfiltered access logger to record information about all client requests, and also additional filtered access logs as described below in "Access Log Filtering".
>
> Do not enable multiple *unfiltered* access loggers, however.
>
> An unfiltered access logger can put significant write load on the disk subsystem where access logs are stored. Every client request results in at least one new log message.

By default, the JSON file-based access logger is enabled. For details about this log publisher, see "Configuring JSON Access Logs".

18.5.1.1. Common ForgeRock Access Logs

OpenDJ servers support the ForgeRock Common Audit event framework, and uses the JSON file handler as the default logger. The log message formats are compatible for all products using the framework. The framework uses transaction IDs to correlate requests as they traverse the platform. This makes it easier to monitor activity and to enrich reports.

Interface stability: Evolving

The ForgeRock Common Audit event framework is built on audit event handlers. Audit event handlers can encapsulate their own configurations. Audit event handlers are the same in each product in the ForgeRock platform. You can plug in custom handlers that comply with the framework without having to upgrade the server.

> **Note**
>
> The ForgeRock Common Audit event framework includes handlers for logging audit event messages to local files and facilities, as well as to remote systems.
>
> Although the ForgeRock Common Audit event framework supports multiple topics, OpenDJ software currently supports handling only access events. OpenDJ software divides access events into `ldap-access` events and `http-access` events.

Common Audit transaction IDs are not recorded by default. In order to record transaction IDs in the access logs, you must configure the OpenDJ server to trust them as described in the sections below.

Common Audit LDAP events have the following format:

```
{
    "eventName": "DJ-LDAP",
    "client": {
        "ip": string,                          // Client IP address
        "port": number                         // Client port number
    },
    "server": {
        "ip": string,                          // Server IP address
        "port": number                         // Server port number
    },
    "request": {                               // LDAP request
        "attrs": [ string ],                   // Requested attributes
        "authType": string,                    // Bind type such as "SIMPLE"
        "connId": number,                      // Connection ID
        "controls": string,                    // Request controls
        "deleteOldRDN": boolean,               // For a modify DN request
        "dn": string,                          // Bind DN
        "filter": string,                      // Search filter
        "idToAbandon": number,                 // ID to use to abandon operation
        "message": string,                     // Localized request message
        "msgId": number,                       // Message ID
        "name": string,                        // Operation name
        "newRDN": string,                      // For a modify DN request
        "newSup": string,                      // For a modify DN request
        "oid": string,                         // Operation name or OID
        "operation": string,                   // Examples: "CONNECT", "BIND", "SEARCH"
        "opType": "sync",                      // Replication operation
        "protocol": "LDAP",
        "runAs": string,                       // Authorization ID
        "scope": string,                       // Search scope such as "sub"
        "version": string                      // Version "2", "3"
    },
    "response": {
        "additionalItems": string              // Additional information
        "controls": string,                    // Response controls
        "elapsedTime": number,                 // Number of time units
        "elapsedTimeUnits": string,            // Time unit such as "MILLISECONDS"
        "failureReason": string,               // Human-readable information
        "maskedMessage": string,               // Real, masked result message
        "maskedResult": string,                // Real, masked result code
        "nentries": number,                    // Number of entries returned
        "reason": string,                      // Reason for disconnect
```

```
      "status": string,                    // "SUCCESSFUL", "FAILED"
      "statusCode": string                 // For example, "0" for success
    },
    "timestamp": string,                   // UTC date
    "transactionId": string,               // Unique ID for the transaction
    "userId": string,                      // User who requested the operation
    "_id": string                          // Unique ID for the operation
}
```

Common Audit HTTP events have the following format:

```
{
    "eventName": "DJ-HTTP",
    "client": {
        "ip": string,                      // Client IP address
        "port": number                     // Client port number
    },
    "server": {
        "ip": string,                      // Server IP address
        "port": number                     // Server port number
    },
    "http": {                              // HTTP request and response
      "request": {
        "secure": boolean,                 // HTTP: false; HTTPS: true
        "method": string,                  // Examples: "GET", "POST", "PUT"
        "path": string,                    // URL
        "queryParameters": map,            // map: { key-string: [ value-string ] }
        "cookies": map                     // map: { key-string: [ value-string ] }
      },
      "response": {
        "headers": map                     // map: { key-string: [ value-string ] }
      }
    },
    "response": {
        "detail": string,                  // Human-readable information
        "elapsedTime": number,             // Number of time units
        "elapsedTimeUnits": string,        // Time unit such as "MILLISECONDS"
        "status": string,                  // "SUCCESSFUL", "FAILED"
        "statusCode": string               // For example, "0" for success
    },
    "timestamp": string,                   // UTC date
    "transactionId": string,               // Unique ID for the transaction
    "trackingIds": [ string ],             // Unique IDs from the transaction context
    "userId": string,                      // User who requested the operation
    "_id": string                          // Unique ID for the operation
}
```

18.5.1.1.1. Configuring JSON Access Logs

A JSON handler sends messages to a JSON format file.

This section includes the following procedures:

• "To Configure JSON LDAP Access Logs"

- "To Enable JSON HTTP Access Logs"

To Configure JSON LDAP Access Logs

The default JSON-based LDAP access log file is `logs/ldap-access.audit.json`. Follow these steps to change the configuration:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs are used to correlate audit events for monitoring and reporting that trace the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal OpenDJ request control. They are sent over HTTP in an HTTP header.

   By default, an OpenDJ server is configured not to trust transaction IDs sent with client application requests. The default transaction ID is used instead. The default transaction ID is zero: `0`.

   To trust transactions, set the advanced global server property, `trust-transaction-ids`, to `true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --trustAll \
    --no-prompt
   ```

2. (Optional) Edit the default access log publisher if necessary.

   The following example applies the default settings:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "Json File-Based Access Logger" \
    --set enabled:true \
    --add "rotation-policy:24 Hours Time Limit Rotation Policy" \
    --add "rotation-policy:Size Limit Rotation Policy" \
    --set "retention-policy:File Count Retention Policy" \
    --trustAll \
    --no-prompt
   ```

To Enable JSON HTTP Access Logs

When you enable the HTTP connection handler as described in "To Set Up REST Access to User Data", also consider enabling JSON-based HTTP access logs.

The default JSON-based HTTP access log file is `logs/http-access.audit.json`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs are used to correlate audit events for monitoring and reporting that trace the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal OpenDJ request control. They are sent over HTTP in an HTTP header.

   By default, an OpenDJ server is configured not to trust transaction IDs sent with client application requests. The default transaction ID is used instead. The default transaction ID is zero: `0`.

   To trust transactions, set the advanced global server property, `trust-transaction-ids`, to `true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --trustAll \
    --no-prompt
   ```

2. Enable the log publisher:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "Json File-Based HTTP Access Logger" \
    --set enabled:true \
    --trustAll \
    --no-prompt
   ```

18.5.1.1.2. Configuring CSV Access Logs

A CSV handler sends messages to a comma-separated variable (CSV) file.

This section includes the following procedures:

- "To Enable CSV LDAP Access Logs"

- "To Enable CSV HTTP Access Logs"

To Enable CSV LDAP Access Logs

The default CSV LDAP access log file is `logs/ldap-access.csv`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs are used to correlate audit events for monitoring and reporting that trace the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal OpenDJ request control. They are sent over HTTP in an HTTP header.

   By default, an OpenDJ server is configured not to trust transaction IDs sent with client application requests. The default transaction ID is used instead. The default transaction ID is zero: `0`.

   To trust transactions, set the advanced global server property, `trust-transaction-ids`, to `true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --trustAll \
    --no-prompt
   ```

2. Create an enabled CSV file access logger with optional rotation and retention policies as in the following example:

```
$ dsconfig \
 create-log-publisher \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
 --type csv-file-access \
 --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" \
 --trustAll \
 --no-prompt
```

When setting the CSV file access log publisher properties, you can set the log directory, but you cannot change the log file name, which contains `ldap-access`.

3. (Optional) If you require tamper-evident logs, prepare a keystore as described in "To Prepare a Keystore for Tamper-Evident Logs". Then enable tamper-evident capability as in the following example:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin-file:config/audit-keystore.pin \
 --trustAll \
 --no-prompt
```

Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.

Be certain to test the performance impact of tamper-evident logging with realistic access patterns for your deployment before enabling the feature in production.

To Enable CSV HTTP Access Logs

If you have enabled the HTTP connection handler as described in "To Set Up REST Access to User Data", you might want to enable CSV-format HTTP access logs.

The default CSV HTTP access log file is `logs/http-access.csv`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs are used to correlate audit events for monitoring and reporting that trace the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal OpenDJ request control. They are sent over HTTP in an HTTP header.

   By default, an OpenDJ server is configured not to trust transaction IDs sent with client application requests. The default transaction ID is used instead. The default transaction ID is zero: `0`.

   To trust transactions, set the advanced global server property, `trust-transaction-ids`, to `true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --trustAll \
    --no-prompt
   ```

2. Create an enabled CSV file HTTP access logger with optional rotation and retention policies as in the following example:

   ```
   $ dsconfig \
    create-log-publisher \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "Common Audit Csv File HTTP Access Logger" \
    --type csv-file-http-access \
    --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" \
    --trustAll \
    --no-prompt
   ```

   When setting the CSV file HTTP access log publisher properties, you can set the log directory, but you cannot change the log file name, which contains `http-access`.

3. (Optional) If you require tamper-evident logs, prepare a keystore as described in "To Prepare a Keystore for Tamper-Evident Logs". Then enable tamper-evident capability as in the following example:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access Logger" \
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin-file:config/audit-keystore.pin \
 --trustAll \
 --no-prompt
```

Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.

Be certain to test the performance impact of tamper-evident logging with realistic access patterns for your deployment before enabling the feature in production.

18.5.1.1.3. Configuring Elasticsearch Access Logs

An Elasticsearch audit event handler sends messages to an Elasticsearch server.

Before you enable the Elasticsearch handler, create a mapping in the Elasticsearch server index for the messages. For a sample index definition, see `/path/to/opendj/config/audit-handlers/elasticsearch-index-setup-example.json`.

To enable the Elasticsearch handler, see "To Enable an Access Logger With an External Configuration". The JSON configuration file for the Elasticsearch handler has the following format:

```
{
  "class": "org.forgerock.audit.handlers.elasticsearch.ElasticsearchAuditEventHandler",
  "config": {
    "name": string,               // Handler name, such as "elasticsearch".
    "topics": [ string, ...],     // LDAP: "ldap-access"; HTTP: "http-access".
    "enabled": boolean,           // Is the handler enabled?
    "connection": {               // (Optional) Connect using default settings.
      "host": string,             // Elasticsearch host. Default: localhost.
      "port": number,             // Elasticsearch host. Default: 9200.
      "useSSL": boolean,          // Connect to Elasticsearch over HTTPS? Default: false.
      "username": string,         // (Optional) User name for HTTP Basic auth.
      "password": string          // (Optional) Password for HTTP Basic auth.
    },
    "indexMapping": {             // (Optional) No mapping specified.
      "indexName": string        // Name of the Elasticsearch index.
    },
    "buffering": {                // (Optional) Default: write each message separately, no buffering.
      "enabled": boolean,         // Buffer messages to be sent? Default: false.
      "maxSize": number,          // Maximum number of buffered events.
      "writeInterval": duration,  // Interval between sending batch of events.
      "maxBatchedEvents": number  // Number of events to send per interval.
    }
  }
}
```

For a sample configuration, see `/path/to/opendj/config/audit-handlers/elasticsearch-config.json-example`.

The `writeInterval` takes a duration.

A duration is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`.

Durations are not case sensitive.

Negative durations are not supported.

The following units can be used in durations:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration

- `zero`, `disabled`: zero-length duration

- `days`, `day`, `d`: days

- `hours`, `hour`, `h`: hours

- `minutes`, `minute`, `min`, `m`: minutes

- `seconds`, `second`, `sec`, `s`: seconds

- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds

- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds

- nanoseconds, nanosecond, nanosec, nanos, nano, ns: nanoseconds

Example: Elasticsearch For Access Log

This example demonstrates logging HTTP access messages to a local Elasticsearch server.

To prepare the example, complete these steps:

1. Install and run an Elasticsearch server on `localhost:9200`.

2. Create an `audit` index in the Elasticsearch server for HTTP audit event messages.

   The following command uses the example index configuration file:

   ```
   $ curl \
    --request POST \
    --header "Content-Type: application/json" \
    --data @/path/to/opendj/config/audit-handlers/elasticsearch-index-setup-example.json \
    http://localhost:9200/audit
   {"acknowledged":true}
   ```

3. Configure OpenDJ servers to enable HTTP access as described in "To Set Up REST Access to User Data".

4. Add a JSON configuration file under for the handler:

   ```
   $ cat /path/to/opendj/config/audit-handlers/elasticsearch-handler.json
   {
     "class": "org.forgerock.audit.handlers.elasticsearch.ElasticsearchAuditEventHandler",
     "config": {
       "name": "elasticsearch",
       "topics": ["http-access"],
       "connection": {
         "useSSL": false,
         "host": "localhost",
         "port": 9200
       },
       "indexMapping": {
         "indexName": "audit"
       },
       "buffering": {
         "enabled": true,
         "maxSize": 10000,
         "writeInterval": "100 ms",
         "maxBatchedEvents": 500
       }
     }
   }
   ```

5. Configure the server to use the Elasticsearch audit handler:

```
$ dsconfig \
create-log-publisher \
--hostname opendj.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--publisher-name "Elasticsearch HTTP Access Log Publisher" \
--type external-http-access \
--set enabled:true \
--set config-file:config/audit-handlers/elasticsearch-handler.json \
--trustAll \
--no-prompt
```

With Elasticsearch and the OpenDJ server running, audit event messages for HTTP requests to the OpenDJ server are sent to Elasticsearch.

The following example requests Babs Jensen's entry:

```
$ curl --user bjensen:hifalutin http://opendj.example.com:8080/api/users/bjensen
```

A search request to Elasticsearch shows the resulting audit event content:

```
$ curl 'localhost:9200/audit/_search?q=*&pretty'
```

See the Elasticsearch documentation for details on searching and search results.

18.5.1.1.4. Configuring JDBC Access Logs

A JDBC handler sends messages to an appropriately configured relational database table.

Before you enable the JDBC handler, create the necessary schema and tables in the target database. See the examples, /path/to/opendj/config/audit-handlers/mysql_tables-example.sql, and /path/to/opendj/config/audit-handlers/oracle_tables-example.sql.

In addition, the JDBC handler depends on the JDBC driver for the database, and HirakiCP. Copy the JDBC driver .jar file for your database, the HirakiCP .jar file for your Java version, and any other dependent libraries required to the /path/to/opendj/extlib/ directory.

To enable the JDBC handler, see "To Enable an Access Logger With an External Configuration". The JSON configuration file for the JDBC handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.jdbc.JdbcAuditEventHandler",
    "config": {
        "name": string,                          // Handler name, such as "jdbc".
```

```
        "topics": array,                   // LDAP: "ldap-access"; HTTP: "http-access".
        "databaseType": string,            // Supported by default: "oracle", "mysql", "h2".
        "enabled": boolean,                // Is the handler enabled?
        "buffering": {                     // (Optional) Default: write each message separately, no
buffering.
            "enabled": boolean,            // Buffer messages to be sent? Default: false.
            "writeInterval": duration,     // Duration; must be > 0 if buffering is enabled.
            "autoFlush": boolean,          // Flush messages automatically? Default: true.
            "maxBatchedEvents": number,    // Maximum messages in prepared statement. Default: 100.
            "maxSize": number,             // Maximum number of buffered messages. Default: 5000.
            "writerThreads": number        // Threads to write buffered messages: Default: 1.
        },
        "connectionPool": {
            "dataSourceClassName": string, // Either set this to the class name of the data source...
            "jdbcUrl": string,             // ...or set this to the JDBC URL to connect to the database.
            "username": string,            // Username to connect to the database.
            "password": string,            // Password to connect to the database.
            "autoCommit": boolean,         // (Optional) Commit transactions automatically? Default: true.
            "connectionTimeout": number,   // (Optional) Milliseconds to wait before timing out. Default:
30,000.
            "idleTimeout": number,         // (Optional) Milliseconds to wait before timing out. Default:
600,000.
            "maxLifetime": number,         // (Optional) Milliseconds thread remains in pool. Default:
1,800,000.
            "minIdle": number,             // (Optional) Minimum connections in pool. Default: 10.
            "maxPoolSize": number,         // (Optional) Maximum number of connections in pool. Default:
10.
            "poolName": string,            // (Optional) Name of connection pool. Default: audit.
            "driverClassName": string      // (Optional) Class name of database driver. Default: null.
        },
        "tableMappings": [                 // Correspondence of message fields to database columns.
            {
                "event": string,           // LDAP: "ldap-access"; HTTP: "http-access".
                "table": string,           // LDAP: "ldapaccess"; HTTP: "httpaccess".
                "fieldToColumn": {         // Map of field names to database column names.
                    "event-field": "database-column" // Event-field takes JSON pointer.
                }
            }
        ]
    }
}
```

For a sample configuration, see `/path/to/opendj/config/audit-handlers/jdbc-config.json-example`.

The `writeInterval` takes a duration.

A duration is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`.

Durations are not case sensitive.

Negative durations are not supported.

The following units can be used in durations:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration

- `zero`, `disabled`: zero-length duration

- `days`, `day`, `d`: days

- `hours`, `hour`, `h`: hours

- `minutes`, `minute`, `min`, `m`: minutes

- `seconds`, `second`, `sec`, `s`: seconds

- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds

- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds

- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

18.5.1.1.5. Configuring JMS Access Logs

A JMS handler is a JMS producer that publishes messages to an appropriately configured Java Message Service.

To enable the JMS handler, see "To Enable an Access Logger With an External Configuration". The JSON configuration file for the JMS handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.jms.JmsAuditEventHandler",
    "config": {
        "name": string,                    // Handler name, such as "jms".
        "enabled": boolean,                // Is the handler enabled?
        "topics": array,                   // LDAP: "ldap-access"; HTTP: "http-access".
        "deliveryMode": string,            // One of "NON_PERSISTENT", "PERSISTENT".
        "sessionMode": string,             // One of "AUTO", "CLIENT", "DUPS_OK".
        "batch": {                         // (Optional) Default: no batching.
            "batchEnabled": boolean,       // Batch messages to publish? Default: false.
            "capacity": number,            // Maximum capacity of publishing queue. Default: 1.
            "maxBatchedEvents": number,    // Maximum events to deliver in single publishing call.
Default: 1.
            "threadCount": number,         // Worker threads to process publishing queue. Default: 1.
            "insertTimeoutSec": number,    // Seconds queue can block before adding new item. Default: 60.
            "pollTimeoutSec": number,      // Seconds worker threads wait for new item. Default: 10.
            "shutdownTimeoutSec": number   // Seconds publisher waits for threads at shutdown. Default:
60.
        },
        "jndi": {                          // (Optional) Default: Use default settings.
            "connectionFactoryName": string, // JNDI name for JMS connection factory. Default:
"ConnectionFactory".
            "topicName": string            // (Optional) Match the value in the context. Default: "audit".
            "contextProperties": {         // JNDI InitialContext properties.
                // These depend on the JNDI provider. See the provider documentation for details.
            }
        }
    }
}
```

For a sample configuration, see `/path/to/opendj/config/audit-handlers/jms-config.json-example`.

18.5.1.1.6. Configuring Splunk Access Logs

A Splunk handler sends messages to an appropriately configured Splunk service.

To enable the Splunk handler, see "To Enable an Access Logger With an External Configuration". The JSON configuration file for the Splunk handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.splunk.SplunkAuditEventHandler",
    "config": {
        "name": string,                     // Handler name, such as "splunk".
        "enabled": boolean,                 // Is the handler enabled?
        "topics": array,                    // LDAP: "ldap-access"; HTTP: "http-access".
        "authzToken": string,               // Splunk authorization token for HTTP requests.
        "buffering": {                      // Required message buffering configuration.
            "maxBatchedEvents": number,     // Maximum messages in prepared statement.
            "maxSize": number,              // Maximum number of buffered messages.
            "writeInterval": duration       // Duration as described below.
        },
        "connection": {                    // (Optional) Default: Use default settings.
            "host": string,                 // Splunk host name. Default: "localhost".
            "port": number,                 // Splunk port number. Default: "8088".
            "useSSL": boolean               // Use secure connection to Splunk? Default: false.
        }
    }
}
```

For a sample configuration, see `/path/to/opendj/config/audit-handlers/splunk-config.json-example`.

The `writeInterval` takes a duration.

A duration is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`.

Durations are not case sensitive.

Negative durations are not supported.

The following units can be used in durations:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration

- `zero`, `disabled`: zero-length duration

- `days`, `day`, `d`: days

- `hours`, `hour`, `h`: hours

- `minutes`, `minute`, `min`, `m`: minutes

- `seconds`, `second`, `sec`, `s`: seconds

- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds

- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds

- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

18.5.1.1.7. Configuring Access Logging to Syslog

A Syslog handler sends messages to the UNIX system log as governed by RFC 5424, *The Syslog Protocol*.

> **Note**
>
> The implementation currently only supports writing *access* messages to Syslog, rather than error messages. As a result, this feature is of limited use in most deployments.

To enable a Syslog handler, see "To Enable an Access Logger With an External Configuration". The JSON configuration file for the Syslog handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.syslog.SyslogAuditEventHandler",
    "config": {
        "name": string,            // Handler name, such as "syslog".
        "enabled": boolean,        // Default: false.
        "topics": array,           // LDAP: "ldap-access"; HTTP: "http-access".
        "protocol": string,        // "TCP" or "UDP".
        "host": string,            // Syslog daemon host, such as localhost; must resolve to IP address.
        "port": number,            // Syslog daemon port number, such as 514; range: 0 to 65535.
        "connectTimeout": number,  // If using TCP, milliseconds to wait before timing out.
        "facility": string,        // Syslog facility to use for event messages.
        "buffering": {             // (Optional) Default: write each message separately, no buffering.
            "enabled": boolean,    // Buffer messages to be sent? Default: false.
            "maxSize": number      // Maximum number of buffered messages. Default: 5000.
        }
    }
}
```

For a sample configuration, see `/path/to/opendj/config/audit-handlers/syslog-config.json-example`.

For additional details, see "Syslog Facility Values".

Syslog Facility Values

Value	Description
`kern`	Kernel messages.
`user`	User-level messages.
`mail`	Mail system.
`daemon`	System daemons.

Value	Description
auth	Security/authorization messages.
syslog	Messages generated internally by syslogd.
lpr	Line printer subsystem.
news	Network news subsystem.
uucp	UUCP subsystem.
cron	Clock daemon.
authpriv	Security/authorization messages.
ftp	FTP daemon.
ntp	NTP subsystem.
logaudit	Log audit.
logalert	Log alert.
clockd	Clock daemon.
local0	Local use 0.
local1	Local use 1.
local2	Local use 2.
local3	Local use 3.
local4	Local use 4.
local5	Local use 5.
local6	Local use 6.
local7	Local use 7.

18.5.1.1.8. Configuring Tamper-Evidence and External Handlers

This section includes the following procedures:

- "To Prepare a Keystore for Tamper-Evident Logs"

- "To Enable an Access Logger With an External Configuration"

To Prepare a Keystore for Tamper-Evident Logs

Tamper-evident logging depends on a public key/private key pair and on a secret key that are stored together in a JCEKS keystore. Follow these steps to prepare the keystore:

1. Create a password for the keystore.

   The following example uses the default file name. If you use a different filename, then you must edit key-store-pin-file property when configuring the log publisher:

```
$ touch /path/to/opendj/config/audit-keystore.pin
$ chmod 600 /path/to/opendj/config/audit-keystore.pin
# Add password in cleartext on the only line in the file:
$ vi /path/to/opendj/config/audit-keystore.pin
```

2. Generate a key pair in the keystore.

   The CSV event handler expects a JCEKS-type keystore with a key alias of `Signature` for the signing key, where the key is generated with the `RSA` key algorithm and the `SHA256withRSA` signature algorithm.

   The following example uses the default file name. If you use a different filename, then you must edit `key-store-file` property when configuring the log publisher:

```
$ keytool \
 -genkeypair \
 -keyalg RSA \
 -sigalg SHA256withRSA \
 -alias "Signature" \
 -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:file /path/to/opendj/config/audit-keystore.pin \
 -keypass:file /path/to/opendj/config/audit-keystore.pin
```

3. Generate a secret key in the keystore.

   The CSV event handler expects a JCEKS-type keystore with a key alias of `Password` for the symmetric key, where the key is generated with the `HmacSHA256` key algorithm and 256-bit key size.

   The following example uses the default file name. If you use a different filename, then you must edit `key-store-file` property when configuring the log publisher:

```
$ keytool \
 -genseckey \
 -keyalg HmacSHA256 \
 -keysize 256 \
 -alias "Password" \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:file /path/to/opendj/config/audit-keystore.pin \
 -keypass:file /path/to/opendj/config/audit-keystore.pin
```

4. Verify that the keystore contains signature and password keys:

```
$ keytool \
 -list \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:file /path/to/opendj/config/audit-keystore.pin
...
signature, <date>, PrivateKeyEntry,
<fingerprint>
password, <date>, SecretKeyEntry,
<fingerprint>
```

To Enable an Access Logger With an External Configuration

An access logger configuration that relies on a JSON configuration lets you use any Common Audit event handler, including customer handlers. The content of the configuration file depends on the audit event handler.

Follow these steps:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs are used to correlate audit events for monitoring and reporting that trace the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal OpenDJ request control. They are sent over HTTP in an HTTP header.

   By default, an OpenDJ server is configured not to trust transaction IDs sent with client application requests. The default transaction ID is used instead. The default transaction ID is zero: `0`.

   To trust transactions, set the advanced global server property, `trust-transaction-ids`, to `true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --trustAll \
    --no-prompt
   ```

2. Create the external JSON configuration file for the handler.

   Base your work on the appropriate template in the `config/audit-handlers` directory.

3. (Optional) If this is a custom access logger provided separately, copy the custom handler .jar file to `/path/to/opendj/lib/extensions`.

4. Create a log publisher configuration for the access log, where the `type` defines whether the log contains messages about LDAP or HTTP requests:

   • For LDAP access logging, create an external access log publisher:

   ```
   $ dsconfig \
    create-log-publisher \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "My External LDAP Access Log Publisher" \
    --type external-access \
    --set enabled:true \
    --set config-file:config/audit-handlers/handler-conf.json \
    --trustAll \
    --no-prompt
   ```

   • For HTTP access logging, create an external HTTP access log publisher:

   ```
   $ dsconfig \
    create-log-publisher \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "My External HTTP Access Log Publisher" \
    --type external-http-access \
    --set enabled:true \
    --set config-file:config/audit-handlers/handler-conf.json \
    --trustAll \
    --no-prompt
   ```

18.5.1.2. Native LDAP Access Logs

For LDAP connection handlers, you can configure a native format access log.

> **Tip**
>
> This format was previously the default for OpenDJ servers. This log format can be useful, for example, if you already have software configured to consume the messages.

This access log uses the File Based Access Log Publisher The default log file is `logs/access`.

The following command enables this LDAP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Access Logger" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

By default, this access log contains a message for each request, and a message for each response, as well as messages for connection and disconnection. You can configure the access log to write messages only on responses by setting the property `log-format:combined`. The setting is useful when filtering messages based on response criteria. It causes the server to log one message per operation, rather than one message for the request and another for the response.

18.5.1.3. Standard HTTP Access Logs

For HTTP requests, you can configure an access logger that uses the Extended Log File Format, which is a W3C working draft.

This access log uses the File Based HTTP Access Log Publisher. The default log file is `logs/http-access`.

The following command enables this HTTP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

The following default fields are shown here in the order they occur in the log file:

`cs-host`

    Client host name

`c-ip`

    Client IP address

`cs-username`

    Username used to authenticate

x-datetime

Completion timestamp for the HTTP request, which you can configure using the `log-record-time-format` property

cs-method

HTTP method requested by the client

cs-uri

URI requested by the client

cs-uri-stem

URL-encoded path requested by the client

cs-uri-query

URL-encoded query parameter string requested by the client

cs-version

HTTP version requested by the client

sc-status

HTTP status code for the operation

cs(User-Agent)

User-Agent identifier

x-connection-id

Connection ID used for OpenDJ internal operations

When using this field to match HTTP requests with internal operations in the LDAP access log, first set the access log advanced property, `suppress-internal-operations`, to `false`. By default, internal operations do not appear in the LDAP access log.

x-etime

Execution time in milliseconds needed by OpenDJ to service the HTTP request

x-transaction-id

ForgeRock Common Audit event framework transaction ID for the request

This defaults to `0` unless you configure the server to trust transaction IDs.

Missing values are replaced with `-`. Tabs separate the fields, and if a field contains a tab character, then the field is surrounded with double quotes. OpenDJ then doubles double quotes in the field to escape them.

The following example shows an excerpt of an HTTP access log with space reformatted:

```
- <client-ip> bjensen  <datestamp> GET  /users/bjensen HTTP/1.1 200 <user-agent> 3  40
- <client-ip> bjensen  <datestamp> GET  /users/scarter HTTP/1.1 200 <user-agent> 4   9
- <client-ip> -        <datestamp> GET  /users/missing HTTP/1.1 401 <user-agent> 5   0
- <client-ip> kvaughan <datestamp> POST /users         HTTP/1.1 200 <user-agent> 6 120
```

You can configure the `log-format` for the access log using the **dsconfig** command.

In addition to the default fields, the following standard fields are supported:

`c-port`

Client port number

`s-computername`

Server name where the access log was written

`s-ip`

Server IP address

`s-port`

Server port number

18.5.1.4. Access Log Filtering

With the default access log configuration (no filtering), for every client application request, the server writes at least one message to its access log. This volume of logging gives you the information to analyze overall access patterns, or to audit access when you do not know in advance what you are looking for.

When you do know what you are looking for, log filtering lets you throttle logging to focus on what you want to see. You specify the criteria for a filtering policy, and apply the policy to a log publisher.

Log filtering policies use the following criteria:

- Client IP address, bind DN, group membership

- Operation type (abandon, add, bind, compare, connect, delete, disconnect, extended operation, modify, rename, search, and unbind)

- Port number

- Protocol used

- Response time

- Result codes (only log error results, for example)

- Search response criteria (number of entries returned, unindexed search, and others)

- Target DN

- User DN and group membership

A log publisher's filtering policy determines whether to include or exclude log messages that match the criteria.

Example: Exclude Administration-Related Messages

A common development troubleshooting technique consists of sending client requests while tailing the access log:

```
$ tail -f /path/to/opendj/logs/ldap-access.audit.json
```

When the control panel is running, or when the **dsconfig** command accesses the configuration, access log messages are written for administrative operations. These messages can prevent you from seeing the messages of interest from client applications.

This example demonstrates how to filter access log messages for administrative connections over LDAPS on port 4444.

Configure access log filtering criteria:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --criteria-name "Exclude LDAPS on 4444" \
 --type generic \
 --set connection-port-equal-to:4444 \
 --set connection-protocol-equal-to:ldaps \
 --trustAll \
 --no-prompt
```

Activate filtering to exclude administrative messages:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set filtering-policy:exclusive \
 --trustAll \
 --no-prompt
```

The publisher filters messages about administrative requests to port 4444.

Example: Audit Configuration Changes

This example demonstrates how to set up an audit log file to track changes to the server configuration. The LDAP representation of the configuration is found under the base DN `cn=config`.

As described in "Server Logs", an audit log is a type of access log that records changes to directory data in LDIF. The change records have timestamped comments with connection and operation IDs, so that you can correlate the changes with messages in access logs.

Create an audit log publisher:

```
$ dsconfig \
 create-log-publisher \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit Log" \
 --type file-based-audit \
 --set enabled:true \
 --set filtering-policy:inclusive \
 --set log-file:logs/config-audit \
 --set rotation-policy:"24 Hours Time Limit Rotation Policy" \
 --set rotation-policy:"Size Limit Rotation Policy" \
 --set retention-policy:"File Count Retention Policy" \
 --trustAll \
 --no-prompt
```

Create log filtering criteria for the logger that matches operations targeting `cn=config`:

```
$ dsconfig \
 create-log-publisher \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit Log" \
 --type file-based-audit \
 --set enabled:true \
 --set filtering-policy:inclusive \
 --set log-file:logs/config-audit \
 --set rotation-policy:"24 Hours Time Limit Rotation Policy" \
 --set rotation-policy:"Size Limit Rotation Policy" \
 --set retention-policy:"File Count Retention Policy" \
 --trustAll \
 --no-prompt
```

The server now writes to the current audit log file, `/path/to/opendj/logs/config-audit`, whenever an administrator changes the server configuration. The following example output show the resulting LDIF that defines the log filtering criteria:

```
# <datestamp>; conn=<id>; op=<id>
dn: cn=Record changes to cn=config,cn=Filtering Criteria,cn=File-Based Server Configuration Audit
 Log,cn=Loggers,cn=config
changetype: add
objectClass: top
objectClass: ds-cfg-access-log-filtering-criteria
cn: Record changes to cn=config
ds-cfg-request-target-dn-equal-to: **,cn=config
ds-cfg-request-target-dn-equal-to: cn=config
createTimestamp: <timestamp>
creatorsName: cn=Directory Manager,cn=Root DNs,cn=config
entryUUID: <uuid>
```

18.5.2. Debug Logs

A *debug log* traces internal server information needed to troubleshoot a problem. Debug logs can grow large quickly, and therefore no debug logs are enabled by default.

For debug logging, you must set a *debug target* to control what gets logged. For details, see "Enabling Debug Logging".

18.5.3. Error Logs

The *errors log* traces server events, error conditions, and warnings, categorized and identified by severity.

Messages in the `logs/errors` file have the following format:

```
[datestamp] category=category severity=severity msgID=ID number msg=message string
```

For lists of severe and fatal error messages by category, see the Log Message Reference.

18.5.4. Replication Repair Logs

The *replication repair log* traces replication events, with entries having the same format as the errors log:

```
[datestamp] category=SYNC severity=severity msgID=ID number msg=message string
```

The replication log does not trace replication operations. Use the external change log instead to get notifications about changes to directory data over protocol, as described in "Change Notification For Your Applications".

18.5.5. Log Rotation and Retention

Each file-based log can be associated with a *log rotation policy*, and a *log retention policy*. The rotation policy specifies when to rotate a log file based on a time, log file age, or log file size. The retention policy specifies whether to retain logs based on the number of logs, their size, or how much free space should be left on the disk.

Rotated logs have a rotation timestamp appended to their name.

You list existing log rotation policies with the **dsconfig list-log-rotation-policies** command, and retention policies with the **dsconfig list-log-retention-policies**:

```
$ dsconfig \
 list-log-rotation-policies \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt
Log Rotation Policy              : Type       : file-size-limit : rotation-interval : time-of-
day
------------------------------------:-----------:-----------------:-------------------:-----------
24 Hours Time Limit Rotation Policy : time-limit : -              : 1 d               : -
7 Days Time Limit Rotation Policy   : time-limit : -             : 1 w               : -
Fixed Time Rotation Policy          : fixed-time : -             : -                 : 2359
Size Limit Rotation Policy          : size-limit : 100 mb        : -                 : -
$ dsconfig \
 list-log-retention-policies \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt
Log Retention Policy            : Type           : disk-space-used : free-disk-space : number-of-
files
------------------------------------:---------------:-----------------:-----------------:----------------
File Count Retention Policy      : file-count    : -              : -               : 10
Free Disk Space Retention Policy : free-disk-space : -            : 500 mb          : -
Size Limit Retention Policy      : size-limit     : 500 mb        : -               : -
```

View the policies that apply for a given log with the **dsconfig get-log-publisher-prop** command. The following example shows that the server keeps 10 access log files, rotating either each day or when the log size reaches 100 MB:

```
$ dsconfig \
 get-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --property retention-policy \
 --property rotation-policy \
 --trustAll \
 --no-prompt
Property         : Value(s)
-----------------:---------------------------------------------------------------
retention-policy : File Count Retention Policy
rotation-policy  : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                 : Policy
```

Use the **dsconfig** command to create, update, delete, and assign log rotation and retention policies. Set the policy that applies to a logger with the **dsconfig set-log-publisher-prop** command.

> **Note**
>
> When using access logs based on the ForgeRock Common Audit event framework, such as the CSV file based or JSON file based access log publishers, you can only configure one of each type of retention or rotation policy.
>
> In other words, you can configure one file count, free disk space, and size limit log retention policy, but not more than one of each. Also, you can configure one fixed time, size limit, and time limit log rotation policy, but not more than one of each.

18.6. Alert Notifications

OpenDJ servers can send notifications of significant server events. Alert notifications are not enabled by default.

The following example enables JMX alert notifications:

```
$ dsconfig \
 set-alert-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "JMX Alert Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

OpenDJ servers can send mail over SMTP instead of JMX notifications. Before you set up the SMTP-based alert handler, specify an SMTP server:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set smtp-server:opendj.example.com \
 --trustAll \
 --no-prompt
$ dsconfig \
 create-alert-handler \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "SMTP Alert Handler" \
 --type smtp \
 --set enabled:true \
 --set message-subject:"OpenDJ Alert, Type: %%alert-type%%, ID: %%alert-id%%" \
 --set message-body:"%%alert-message%%" \
 --set recipient-address:kvaughan@example.com \
 --set sender-address:opendj@example.com \
 --trustAll \
 --no-prompt
```

Alert Types

OpenDJ servers use the following alert types. For alert types that indicate server problems, check `logs/errors` for details:

org.opends.server.AccessControlDisabled

The access control handler has been disabled.

org.opends.server.AccessControlEnabled

The access control handler has been enabled.

org.opends.server.authentiation.dseecompat.ACIParseFailed

The dseecompat access control subsystem failed to correctly parse one or more ACI rules when the server first started.

org.opends.server.BackendRunRecovery

The pluggable backend has thrown a `RunRecoveryException`. The server needs to be restarted.

org.opends.server.CannotCopySchemaFiles

A problem has occurred while attempting to create copies of the existing schema configuration files before making a schema update, and the schema configuration has been left in a potentially inconsistent state.

org.opends.server.CannotRenameCurrentTaskFile

The server is unable to rename the current tasks backing file in the process of trying to write an updated version.

org.opends.server.CannotRenameNewTaskFile

The server is unable to rename the new tasks backing file into place.

org.opends.server.CannotScheduleRecurringIteration

The server is unable to schedule an iteration of a recurring task.

org.opends.server.CannotWriteConfig

The server is unable to write its updated configuration for some reason and therefore the server may not exhibit the new configuration if it is restarted.

org.opends.server.CannotWriteNewSchemaFiles

A problem has occurred while attempting to write new versions of the server schema configuration files, and the schema configuration has been left in a potentially inconsistent state.

org.opends.server.CannotWriteTaskFile

The server is unable to write an updated tasks backing file for some reason.

org.opends.server.DirectoryServerShutdown

The server has begun the process of shutting down.

org.opends.server.DirectoryServerStarted

The server has completed its startup process.

org.opends.server.DiskFull

Free disk space has reached the full threshold.

Default is 20 MB.

org.opends.server.DiskSpaceLow

Free disk space has reached the low threshold.

Default is 100 MB.

org.opends.server.EnteringLockdownMode

The server is entering lockdown mode, wherein only root users are allowed to perform operations and only over the loopback address.

org.opends.server.LDAPHandlerDisabledByConsecutiveFailures

Consecutive failures have occurred in the LDAP connection handler and have caused it to become disabled.

org.opends.server.LDAPHandlerUncaughtError

Uncaught errors in the LDAP connection handler have caused it to become disabled.

org.opends.server.LDIFBackendCannotWriteUpdate

An LDIF backend was unable to store an updated copy of the LDIF file after processing a write operation.

org.opends.server.LDIFConnectionHandlerIOError

The LDIF connection handler encountered an I/O error that prevented it from completing its processing.

org.opends.server.LDIFConnectionHandlerParseError

The LDIF connection handler encountered an unrecoverable error while attempting to parse an LDIF file.

org.opends.server.LeavingLockdownMode

The server is leaving lockdown mode.

org.opends.server.ManualConfigEditHandled

The server detects that its configuration has been manually edited with the server online and those changes were overwritten by another change made through the server. The manually edited configuration will be copied to another location.

org.opends.server.ManualConfigEditLost

The server detects that its configuration has been manually edited with the server online and those changes were overwritten by another change made through the server. The manually edited configuration could not be preserved due to an unexpected error.

org.opends.server.replication.UnresolvedConflict

Multimaster replication cannot resolve a conflict automatically.

org.opends.server.UncaughtException

A server thread has encountered an uncaught exception that caused that thread to terminate abnormally. The impact that this problem has on the server depends on which thread was impacted and the nature of the exception.

org.opends.server.UniqueAttributeSynchronizationConflict

A unique attribute conflict has been detected during synchronization processing.

org.opends.server.UniqueAttributeSynchronizationError

An error occurred while attempting to perform unique attribute conflict detection during synchronization processing.

Chapter 19
Tuning Servers For Performance

This chapter suggests ways to measure and improve directory service performance. In this chapter you will learn to:

- Define directory service performance goals operationally in accordance with the needs of client applications

- Identify constraints that might limit achievable performance goals

- Design and execute appropriate performance tests with the help of OpenDJ command-line tools

- Adjust OpenDJ and system settings to achieve performance goals

Server tuning refers to the art of adjusting server, JVM, and system configuration to meet the service-level performance requirements of directory clients. In the optimal case you achieve service-level performance requirements without much tuning at all, perhaps only setting JVM runtime options when installing an OpenDJ server.

If you are reading this chapter, however, you are probably not facing an optimal situation. Instead you are looking for trade-offs that maximize performance for clients given the constraints of your deployment.

19.1. Defining Performance Requirements and Constraints

Your key performance requirement is most likely to satisfy your users or customers with the resources available to you. Before you can solve potential performance problems, define what those users or customers expect, and determine what resources you will have to satisfy their expectations.

19.1.1. Service-Level Agreements

Service-level agreement (SLA) is a formal name for what directory client applications and the people who run them expect from your service in terms of performance.

SLAs might cover many aspects of the directory service. Whether or not your SLA is formally defined, you ought to know what is expected, or at least what you provide, in the following four areas:

- Directory service *response times*

Directory service response times range from less than a millisecond on average across a low latency connection on the same network to however long it takes your network to deliver the response. More important than average or best response times is the response time distribution, because applications set timeouts based on worst case scenarios. For example, a response time performance requirement might be defined as, *Directory response times must average less than 10 milliseconds for all operations except searches returning more than 10 entries, with 99.9% of response times under 40 milliseconds.*

• Directory service *throughput*

Directory service throughput can range up to many thousands of operations per second. In fact there is no upper limit for read operations such as searches, because only write operations must be replicated. To increase read throughput, simply add additional replicas. More important than average throughput is peak throughput. You might have peak write throughput in the middle of the night when batch jobs update entries in bulk, and peak binds for a special event or first thing Monday morning. For example, a throughput performance requirement might be expressed as, *The directory service must sustain a mix of 5,000 operations per second made up of 70% reads, 25% modifies, 3% adds, and 2% deletes.*

Even better is to mimic the behavior of key operations for performance testing, so that you understand the patterns of operations in the throughput you need to provide.

• Directory service *availability*

OpenDJ software is designed to let you build directory services that are basically available, including during maintenance and even upgrade of individual servers. Yet, in order to reach very high levels of availability, you must make sure not only that the software is designed for availability, but also that your operations execute in such a way as to preserve availability. Availability requirements can be as lax as best effort, or as stringent as 99.999% or more uptime.

Replication is the OpenDJ feature that allows you to build a highly available directory service.

• Directory service administrative support

Be sure to understand how you support your users when they run into trouble. While directory services can help you turn password management into a self-service visit to a web site, some users still need to know what they can expect if they need your help.

Creating an SLA, even if your first version consists of guesses, helps you reduce performance tuning from an open-ended project to a clear set of measurable goals for a manageable project with a definite outcome.

19.1.2. Available Resources

With your SLA in hand, inventory the server, networks, storage, people, and other resources at your disposal. Now is the time to estimate whether it is possible to meet the requirements at all.

If, for example you are expected to serve more throughput than the network can transfer, maintain high-availability with only one physical machine, store 100 GB of backups on a 50 GB partition, or provide 24/7 support all alone, no amount of tweaking available resources is likely to fix the problem.

When checking that the resources you have at least theoretically suffice to meet your requirements, do not forget that high availability in particular requires at least two of everything to avoid single points of failure. Be sure to list the resources you expect to have, when and how long you expect to have them, and why you need them. Also make note of what is missing and why.

In addition to the suggestions in this section, also read "Choosing Hardware" in the *Release Notes*.

19.1.2.1. Server Hardware Recommendations

OpenDJ servers are pure Java applications, making them very portable. OpenDJ servers tend to perform best on single-board, x86 systems due to low memory latency.

19.1.2.2. Advice Concerning Storage

High-performance storage is essential for handling high-write throughput. When the database stays fully cached in memory, directory read operations do not result in disk I/O. Only writes result in disk I/O. You can further improve write performance by using solid-state disks for persistent storage, or for file system cache.

> **Important**
>
> OpenDJ directory servers are designed to work with *local storage* for database backends. *Do not use network file systems, such as NFS, where there is no guarantee that a single process has access to files.*
>
> Storage area networks (SANs) and attached storage are fine for use with an OpenDJ directory server.

Regarding database size on disk, sustained write traffic can cause the database to grow to more than twice its initial size on disk. This is normal behavior. The size on disk does not impact the DB cache size requirements.

In order to avoid directory database file corruption after crashes or power failures on Linux systems, enable file system write barriers and make sure that the file system journaling mode is ordered. For details on how to enable write barriers and how to set the journaling mode for data, see the options for your file system in the **mount** command manual page.

19.2. Testing Performance

Even if you do not need high availability, you still need two of everything, because your test environment needs to mimic your production environment as closely as possible if you want to avoid unwelcome surprises.

In your test environment, you set up an OpenDJ server as you will later in production, and then conduct experiments to determine how to best meet the requirements defined in the SLA.

Use the **makeldif** command, described in makeldif(1) in the *Reference*, to generate sample data that match what you expect to find in production.

The following command-line tools help with basic performance testing:

- The **addrate** command, described in addrate(1) in the *Reference*, measures add and delete throughput and response time.

- The **authrate** command, described in authrate(1) in the *Reference*, measures bind throughput and response time.

- The **modrate** command, described in modrate(1) in the *Reference*, measures modification throughput and response time.

- The **searchrate** command, described in searchrate(1) in the *Reference*, measures search throughput and response time.

All these commands show you information about the response time distributions, and allow you to perform tests at specific levels of throughput.

If you need additional precision when evaluating response times, use the global configuration setting `etime-resolution`, to change elapsed processing time resolution from milliseconds (default) to nanoseconds:

```
$ dsconfig \
  set-global-configuration-prop \
  --hostname opendj.example.com \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --set etime-resolution:nanoseconds \
  --trustAll \
  --no-prompt
```

The `etime`, recorded in the server access log, indicates the elapsed time to process the request, starting from the moment the decoded operation is available to be processed by a worker thread.

19.3. Tweaking OpenDJ Performance

When your tests show that OpenDJ performance is lacking even though you have the right underlying network, hardware, storage, and system resources in place, you can tweak OpenDJ performance in a number of ways. This section covers the most common tweaks.

19.3.1. Setting Maximum Open Files

An OpenDJ server needs to be able to open many file descriptors, especially when handling thousands of client connections. Linux systems in particular often set a limit of 1024 per user, which is too low to handle many client connections to an OpenDJ server.

When setting up an OpenDJ server for production use, make sure the server can use at least 64K (65536) file descriptors. For example, when running the server as user `opendj` on a Linux system that uses `/etc/security/limits.conf` to set user level limits, you can set soft and hard limits by adding these lines to the file:

```
opendj soft nofile 65536
opendj hard nofile 131072
```

The example above assumes the system has enough file descriptors available overall. You can check the Linux system overall maximum as follows:

```
$ cat /proc/sys/fs/file-max
204252
```

19.3.2. Java Settings

Default Java settings let you evaluate OpenDJ servers using limited system resources. If you need high performance for production system, test with the following JVM options. These apply to the Sun/Oracle JVM.

> **Tip**
>
> To apply JVM settings for your server, edit `config/java.properties`:

`-server`

> Use the C2 compiler and optimizer (HotSpot Server VM).

`-d64`

> Use this option on 64-bit systems for heaps larger than 3.5 GB.

`-Xms, -Xmx`

> Set both minimum and maximum heap size to the same value to avoid resizing. Leave space for the entire DB cache and more.
>
> Use at least a 2 GB heap (`-Xms2G -Xmx2G`) unless your data set is small.

`-Xmn`

> Set the new generation size between 1-4 GB for high-throughput deployments, but leave enough overall JVM heap to avoid overlaps with the space used for DB cache.

`-XX:MaxTenuringThreshold=1`

> Force an OpenDJ server to only create objects that have either a short lifetime, or a long lifetime.

`-XX:NewSize`

> If an OpenDJ server handles high throughput, set the new generation size large enough for the JVM to avoid promoting short-lived objects into the old generation space (`-XX:NewSize=512M`).

-XX:+UseConcMarkSweepGC

> The CMS garbage collector tends to give the best performance characteristics with the lowest garbage collection pause times.
>
> Consider using the G1 garbage collector only if CMS performance characteristics do not fit your deployment, and testing shows G1 performs better.

-XX:+UseCompressedOops

> Set this option when you have a 64-bit JVM, and `-Xmx` less than 32 GB. Java object pointers normally have the same size as native machine pointers. If you run a small 64-bit JVM, then compressed object pointers can save space.

-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps

> Use these options when diagnosing JVM tuning problems. You can turn them off when everything is running smoothly.

19.3.3. Data Storage Settings

By default, OpenDJ servers compress attribute descriptions and object class sets to reduce data size. This is called compact encoding.

By default, OpenDJ servers do not, however, compress entries stored in its backend database. If your entries hold values that compress well—such as text— you can gain space by setting the backend property `entries-compressed` to `true` before you (re-)import data from LDIF. With `entries-compressed: true` The OpenDJ server compresses entries before writing them to the database:

```
$ dsconfig \
 set-backend-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set entries-compressed:true \
 --trustAll \
 --no-prompt
$ import-ldif \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --ldifFile Example.ldif \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --start 0 \
 --trustAll
```

An OpenDJ directory server does not proactively rewrite all entries in the database after you change the settings. Instead, to force an OpenDJ server to compress all entries, import the data from LDIF.

19.3.4. LDIF Import Settings

You can tweak OpenDJ servers to speed up import of large LDIF files.

By default, the temporary directory used for scratch files is `import-tmp` under the directory where you installed the OpenDJ server. Use the **import-ldif** command, described in import-ldif(1) in the *Reference*, with the `--tmpdirectory` option to set this directory to a `tmpfs` file system, such as `/tmp`.

If you are certain your LDIF contains only valid entries with correct syntax, because the LDIF was exported with all checks active, for example, you can skip schema validation. Use the `--skipSchemaValidation` option with the **import-ldif** command.

19.3.5. Database Cache Settings

Database cache size is, by default, set as a percentage of the JVM heap by using the backend property `db-cache-percent`. Alternatively, you use the backend property `db-cache-size`, to set the size. If you set up multiple database backends, the total percent of JVM heap used must remain less than 100, and must leave space for other uses. Default settings work for servers with one user data backend JVM heaps up to 2 GB. For heaps larger than 2 GB, you can allocate a larger percentage of heap space to DB cache.

Depending on the size of your database, you have a choice to make about database cache settings:

• By caching the entire database in the JVM heap, you can get more deterministic response times and limit disk I/O. Yet, caching the whole DB can require a very large JVM.

• By allowing file system cache to hold the portion of database that does not fit in the DB cache, you trade less deterministic and slightly slower response times for a smaller JVM heap. How you configure the file system cache depends on your operating system.

19.3.6. Caching Large, Frequently Used Entries

OpenDJ servers implement an entry cache designed for deployments with a few large entries that are regularly updated or accessed. The common use case is a deployment with a few large static groups that are updated or accessed regularly. An entry cache is used to keep such groups in memory in a format that avoids the need to constantly read and deserialize the large entries.

When configuring an entry cache, take care to include only the entries that need to be cached by using the configuration properties `include-filter` and `exclude-filter`. The memory devoted to the entry cache is not available for other purposes.

The following example adds a Soft Reference entry cache to hold entries that match the filter `(ou=Large Static Groups)`. A Soft Reference entry cache allows cached entries to be released if the JVM

is running low on memory. A Soft Reference entry cache has no maximum size setting, so the number of entries cached is limited only by the `include-filter` and `exclude-filter` settings:

```
$ dsconfig \
 create-entry-cache \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --cache-name "Large Group Entry Cache" \
 --type soft-reference \
 --set cache-level:1 \
 --set include-filter:"(ou=Large Static Groups)" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

The entry cache configuration takes effect when the entry cache is enabled.

19.3.7. Logging Settings

Debug logs trace the internal workings of OpenDJ servers, and generally should be used sparingly. Be particularly careful when activating debug logging in high performance deployments.

In general, leave other logs active for production environments to help troubleshoot any issues that arise.

For OpenDJ servers handling very high throughput, however, such as 100,000 operations per second or more, the access log constitute a performance bottleneck. Each client request results in at least one access log message. Consider disabling the access log in such cases.

The following command disables the JSON-based LDAP access logger described in "Configuring JSON Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

The following command disables the access logger described in "Standard HTTP Access Logs":

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

Chapter 20
Securing and Hardening Servers

When you deploy OpenDJ servers in production, there are specific precautions you should take to minimize risks. This chapter recommends the key precautions to take. In this chapter you will learn to:

- Set up a special system account for the OpenDJ server, and appropriately protect access to server files

- Enforce use of the latest Java security updates

- Enable only directory services that are actually used

- Use appropriate log configuration, global access control, password storage, and password policy settings

- Avoid overuse of the default directory root user account

- Use appropriate global access control settings

- Secure connections to the directory

After following the recommendations in this chapter, make sure that you test your installation to verify that it behaves as expected before putting the server into production.

20.1. Setting Up a System Account for an OpenDJ Server

Do not run an OpenDJ server as the system superuser (root). When applications run as superuser, the system effectively does not control their actions. When running the server as superuser, a bug in the server could affect other applications or the system itself.

After setting up a system account for the server, and using that account only to run the server, you can use system controls to limit user access.

The user running the server must have access to use the configured ports. Make sure you configure the system to let the user access privileged ports such as 389 and 636 if necessary. Make sure you configure the firewall to permit access to the server ports.

The user running the server must have access to all server files, including configuration files, data files, log files, keystores, truststores and their password files, and other files. By default, OpenDJ

software lets users in the same group as the user running the server read server files, though not directory data files.

The user running the server does not, however, need access to login from a remote system or to perform actions unrelated to the directory service.

Set up the user account to prevent other users from reading configuration files. On UNIX, set an appropriate umask such as `027` to prevent users in other groups from accessing server files. On Windows, use file access control to do the same. Do consider letting all users to run command-line tools. What a user can do with tools depends on server access control mechanisms. For details, see "Setting Appropriate File Permissions" in the *Security Guide*.

You can create a UNIX service script to start the server at system startup and stop the server at system shutdown by using the **create-rc-script** command. For details see create-rc-script(1) in the *Reference*.

You can use the **windows-service** command to register an OpenDJ server as a Windows service. For details, see windows-service(1) in the *Reference*.

20.2. Using Java Security Updates

Security updates are regularly released for the Java runtime environment.

Make sure that your operational plans provide for deploying Java security updates to systems where you run OpenDJ software.

When you install an OpenDJ server, the path to the Java runtime environment is saved in the configuration. The server continues to use that Java version until you change the configuration as described below.

To Apply Java Security Updates

Follow these steps each time an update to the Java runtime environment changes the Java home path where the runtime environment is installed:

1. (Optional) If the previous Java update used an unlimited strength cryptography policy, install the policy for the updated Java runtime environment as described in "Using Unlimited Strength Cryptography" in the *Security Guide*.

2. (Optional) If an OpenDJ server relies on any CA certificates that were added to the Java runtime environment truststore, `$JAVA_HOME/Home/jre/lib/security/cacerts`, for the previous update, add them to the truststore for the update Java runtime environment.

3. Edit the `default.java-home` setting in the file `config/java.properties` to use the new path.

    The setting should reflect the updated Java home:

```
default.java-home=/path/to/updated/java/jre
```

4. Restart the OpenDJ server to use the updated Java runtime environment:

```
$ stop-ds --restart
```

20.3. Only Enable Necessary Services

By default, an OpenDJ server enables an LDAP connection handler and an administration connector. If the LDAP connection handler is not used, either because only LDAPS is used or because applications access directory data only over HTTPS, then set the LDAP connection handler property to `enabled:false` by using the **dsconfig set-connection-handler-prop** command.

Likewise, if you have enabled other connection handlers that are not used, you can also disable them by using the **dsconfig** command. Use the **status** command to check which connection handlers are enabled.

20.4. Configure Logging Appropriately

By default, an OpenDJ server writes log messages to files when an error is encountered and when the server is accessed. Access logs tend to be much more intensively updated than error logs. You can also configure debug logging, generally too verbose for continuous use in production, and audit logging, which uses the access log mechanism to record changes. Debug and audit logs are not enabled by default. For details, see "Server Logs".

The default OpenDJ server error log levels and log rotation and retention policies are set to prevent the logs from harming performance or filling up the disk while still making it possible to perform basic troubleshooting. If you must set a more verbose error log level or if you must activate debug logging on a production system for more advanced troubleshooting, be aware that extra logging can negatively impact performance and generate large files on heavily used servers. When finished troubleshooting, reset the log configuration for more conservative logging.

The audit log of an OpenDJ server is not for security audits. Instead it records changes in LDIF. The audit log is intended to help you as server administrator to diagnose problems in the way applications change directory data. For change notification as a service, use the external change log instead. For details about the external change log, see "Change Notification For Your Applications".

20.5. Limit Use of the cn=Directory Manager Account

Directory root DN accounts are stored in the server configuration under `cn=Root DNs,cn=config`. In order to bootstrap the system, the default root DN administrator, `cn=Directory Manager`, is not subject

to access control and has privileges to perform almost every administrative operation, including changing privileges.

Use this account like you use the superuser (root) account on UNIX or the Administrator account on Windows: Use it only when you must.

Instead of allowing other applications to perform operations as the root DN administrator `cn=Directory Manager`, either create alternative root DN administrators with limited privileges, or explicitly assign directory administrator rights to specific accounts.

When creating alternative root DN administrators, you can limit their inherited privileges to prevent them from inheriting `bypass-acl` and `privilege-change` privileges. For an example of how to do this see "To Add Privileges for a Group of Administrators".

To explicitly assign rights to specific accounts, create a directory administrator group and add administrators as members. Use the group to assign privileges to the administrators. For details see "To Add Privileges for a Group of Administrators". Create multiple administrator groups if necessary for your deployment.

In both cases, explicitly set up access control instructions (ACIs) to allow administrators to perform administrative actions. For details see "*Configuring Privileges and Access Control*". This prevents administrators from accidentally or intentionally overstepping their authority when managing OpenDJ servers and directory data, and you make it easier to audit what administrators can do.

20.6. Reconsider Default Global Access Control

Global ACIs or access policies are defined in the server configuration. Global access settings apply together with ACIs in the user data.

You can set up a server to apply the recommendations in this section by using the **setup** command option, `--productionMode`.

When you set up a server without using the `--productionMode` option, default global access control settings allow applications to:

- Read the root DSE

- Read server LDAP schema

- Read directory data anonymously

- Modify one's own entry

- Request extended operations and operations with certain controls

For details, see "Default Global ACIs".

If the default global access control settings do not match your requirements, make sure you change them on each server as the server configuration data is not replicated. Global ACIs have the same syntax as ACIs in the directory data. Global access policies are entries in the server configuration. For details about access control settings, see "*Configuring Privileges and Access Control*".

Default global access control settings can and often do change between releases. Review the release notes when upgrading to a new release. For details, see "*Compatibility*" in the *Release Notes*.

Generally it is appropriate to allow anonymous applications to read the root DSE, and to request the StartTLS extended operation over a cleartext connection, even if read access to most directory data requires authorization. The operational attributes on the root DSE indicate the server capabilities, allowing applications to discover interactively how to use the server. The StartTLS extended operation lets an application initiate a secure session starting on a port that does not require encryption.

Authenticated applications should be allowed to read schema operational attributes. LDAP schema operational attributes describe the data stored in the directory. An application that can read schema attributes and check that changes to directory data respect the LDAP schema before sending an update request.

To Minimize Global ACIs

Follow these steps to minimize global ACIs:

1. Remove existing global ACIs to prevent all access.

   ```
   $ dsconfig \
    set-access-control-handler-prop \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --reset global-aci \
    --trustAll \
    --no-prompt
   ```

2. Allow limited global access for essential operations.

   This example allows the following limited access:

   • Authenticated users can request the ForgeRock Transaction ID control, which has OID `1.3.6.1.4.1.36733.2.1.5.1`.

   Other components in the ForgeRock platform use this control to share transaction IDs for common access logging.

   If you do not use common access logging, you can skip adding the global ACI for `Transaction ID control access`.

- Anonymous users can request the Get Symmetric Key extended operation, which has OID `1.3.6.1.4.1.26027.1.6.3`, and the StartTLS extended operation, which has OID `1.3.6.1.4.1.1466.20037`.

  An OpenDJ server requires Get Symmetric Key extended operation access to create and share secret keys for encryption.

  Directory client applications must be able to use the StartTLS operation to initiate a secure connection with an LDAP connection handler. This must be available to anonymous users so that applications can initiate a secure connection before sending bind credentials to authenticate, for example.

  If the directory deployment does not support StartTLS, then remove `|| 1.3.6.1.4.1.1466.20037` from the global ACI for `Anonymous extended operation access`.

- Anonymous and authenticated users can read information about the LDAP features that OpenDJ servers support according to the global ACI named `User-Visible Root DSE Operational Attributes`.

  This exposes metadata publicly for the following attributes:

`namingContexts`

   The base DNs for user data

`supportedAuthPasswordSchemes`

   Supported `authPassword` storage schemes for pre-encoded passwords

`supportedControl`

   Supported LDAP controls by OID

`supportedExtension`

   Supported LDAP extended operations by OID

`supportedFeatures`

   Supported optional LDAP features by OID

`supportedLDAPVersion`

   Supported LDAP versions

`supportedSASLMechanisms`

   Supported SASL mechanisms

`supportedTLSCiphers`

   Supported cipher suites for transport layer security

supportedTLSProtocols

> Supported protocols for transport layer security

vendorName

> Name of the LDAP server implementer

vendorVersion

> Version of the LDAP server implementation

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"1.3.6.1.4.1.36733.2.1.5.1\")\
(version 3.0; acl \"Transaction ID control access\"; allow(read)\
userdn=\"ldap:///all\";)" \
 --add global-aci:"(extop=\"1.3.6.1.4.1.26027.1.6.3 || 1.3.6.1.4.1.1466.20037\") \
(version 3.0; acl \"Anonymous extended operation access\";\
allow(read) userdn=\"ldap:///anyone\";)" \
 --add global-aci:"(target=\"ldap:///\")(targetscope=\"base\")\
(targetattr=\"objectClass||namingContexts||supportedAuthPasswordSchemes\
||supportedControl||supportedExtension||supportedFeatures||supportedLDAPVersion\
||supportedSASLMechanisms||supportedTLSCiphers||supportedTLSProtocols||\
vendorName||vendorVersion\")(version 3.0;\
acl \"User-Visible Root DSE Operational Attributes\";\
allow (read,search,compare) userdn=\"ldap:///anyone\";)" \
 --add global-aci:"(target=\"ldap:///cn=schema\")(targetscope=\"base\")\
(targetattr=\"objectClass||attributeTypes||dITContentRules||dITStructureRules\
||ldapSyntaxes||matchingRules||matchingRuleUse||nameForms||objectClasses\")\
(version 3.0; acl \"User-Visible Schema Operational Attributes\";\
allow (read,search,compare) userdn=\"ldap:///all\";)" \
 --trustAll \
 --no-prompt
```

3. Add or update global ACIs as required by known client applications.

   Work with your partners and with client application documentation for details.

   For example, ForgeRock Access Management servers might require access to update schema definitions under `cn=schema`, and to use the Persistent Search control.

   The following capabilities are useful with the REST to LDAP gateway:

Access to Read Entry controls

By default, the OpenDJ REST to LDAP gateway uses the Pre-Read (OID: `1.3.6.1.1.13.1`) and Post-Read (OID: `1.3.6.1.1.13.2`) controls to read an entry before it is deleted, or to read an entry after it is added or modified.

To determine whether access to request the controls is required by the deployment, check the configuration for `readOnUpdatePolicy` as described in "Gateway REST2LDAP Configuration File" in the *Reference*.

Access to the Subtree Delete control

By default, the OpenDJ REST to LDAP gateway uses the LDAP Subtree Delete (OID: `1.2.840 .113556.1.4.805`) control for delete operations.

To determine whether access to request the controls is required by the deployment, check the configuration for `useSubtreeDelete` as described in "Gateway REST2LDAP Configuration File" in the *Reference*.

Access to the Permissive Modify control

By default, the OpenDJ REST to LDAP gateway uses the LDAP Permissive Modify (OID: `1.2 .840.113556.1.4.1413`) control for LDAP modify operations resulting from patch and update operations.

To determine whether access to request the controls is required by the deployment, check the configuration for `usePermissiveModify` as described in "Gateway REST2LDAP Configuration File" in the *Reference*.

The following example adds control access for OpenDJ REST to LDAP gateway with a default configuration:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"1.3.6.1.1.13.1||1.3.6.1.1.13.2\
 ||1.2.840.113556.1.4.805||1.2.840.113556.1.4.1413\")\
 (version 3.0; acl \"REST to LDAP control access\"; allow(read) userdn=\"ldap:///all\";)" \
 --trustAll \
 --no-prompt
```

20.7. Protect Network Connections

Server protocols like LDAP, HTTP, JMX, and replication rely on transport layer security to protect network connections. For evaluation and initial testing, you might find it useful to inspect network messages without decrypting them. For final testing and production environments, secure the connections.

Transport layer security depends on public key infrastructure when negotiating encryption. OpenDJ servers have keystores for handling their key pairs and public key certificates. For details, see "*Changing Server Certificates*".

An OpenDJ server can simplify installation by self-signing certificates for server key pairs. Self-signed certificates are not recognized by applications until you add them to the application's truststore. This is not a problem when you control both the service and the applications. Self-signed certificates are generally fine even in production systems for administrative and replication connections not used by other applications. For connection handlers that primarily serve applications you do not control, have the server public key certificate signed by a well-known CA so that the applications can recognize the certificate by default. For details on setting up connection handlers for secure communications, see "*Configuring Connection Handlers*".

You can use an access control setting to require secure communications for most operations. Keep a global access control setting that allows anonymous access to request the StartTLS extended operation. When using ACIs, for all operations other than requesting StartTLS, use ACIs whose subject sets `authmethod = ssl`, and also sets `ssf` appropriately. When using global access control policies, use a `connection-minimum-ssf` setting that enforces use of transport layer security, such as 128 or 256.

A security strength factor (`ssf`) is set when the server negotiates connection security with a client application. The `ssf` setting in an ACI subject indicates acceptable security strength factors for the target operation. The server can then check whether the security strength factor for the connection is acceptable according to ACIs that apply. The `ssf` setting in an ACI takes an integer between 0 and 1024. `ssf = 0` (or not set) means cleartext is acceptable. `ssf = 1` calls for integrity protection, meaning the connection should prevent messages from being corrupted between the sender and the receiver. `ssf >= ` *integer* where *integer* is two or more calls for integrity and confidentiality protection. Confidential messages are encrypted. Integers larger than one reflect the effective key size of the cipher negotiated between an OpenDJ server and the LDAP client application. With the `ssf` setting, the aim is to achieve a balance. If not set, or set too low, the server and client can negotiate a connection that is not secure. If set too high, the server and some clients might not be able to negotiate connection settings at all.

When an OpenDJ server and a client application negotiate connection security, they must agree on a security protocol and cipher suite. By default, an OpenDJ server supports all the SSL and TLS protocols and the cipher suites supported by the underlying Java virtual machine. The list can include protocols and ciphers that are not secure enough for the production environment. You can limit the security protocols and ciphers to those that are secure enough. For an example of how to change the settings for a connection handler, see "TLS Protocols and Cipher Suites". You can also change the settings on the administration connector with the **dsconfig set-administration-connector-prop**

command, and change the settings for replication by changing the crypto manager settings with the
dsconfig set-crypto-manager-prop command.

20.8. Use Appropriate Password Storage and Password Policies

Make sure you keep passwords secret in production. The server configuration includes files that hold passwords. Command-line tools allow users to provide password credentials. Passwords are also stored in directory data. This section looks at how to protect passwords in each situation.

20.8.1. Passwords in Configuration Files

An OpenDJ server stores passwords in configuration files.

The `config.ldif` file stores hashes of the passwords for root DN users, such as `cn=Directory Manager`. Likewise for replicated servers the `admin-backend.ldif` file stores a password hash for the global administrator, such as `cn=admin,cn=Administrators,cn=admin data`. By default the password storage algorithm is Salted SHA512, a salted form of the 512-bit SHA-2 message digest algorithm. Permissions on the current copy of the file make it readable and writable only by the user running the server. A backup copy of the version used for the latest successful server startup, `config.ldif.startok`, can be readable to other users depending on the UNIX umask or Windows access control. Use a storage scheme that protects the passwords in server configuration files.

By default, an OpenDJ server stores passwords for keystores and truststores in configuration files with `.pin` extensions. These files contain the cleartext, randomly generated passwords. Keep the PIN files readable and writable only by the user running the server. Alternatively, you can use the **dsconfig** command to configure the server to store keystore and truststore passwords in environment variables or Java properties if your procedures make these methods more secure in production. The settings to change are those of the Key Manager Providers and Trust Manager Providers.

20.8.2. Passwords as Command-Line Arguments

OpenDJ commands supply credentials for any operations that are not anonymous. Password credentials can be supplied as arguments such as the `--bindPassword` *password* option shown in many of the examples in the documentation. The passwords for keystores and truststores are handled in the same way. This is not recommended in production as the password appears in the command. Passwords can also be supplied interactively by using a `-` in the commands, as in `--bindPassword -`. The following example demonstrates a password supplied interactively:

```
$ ldapsearch \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword - \
 --baseDN cn=config \
 "(cn=Directory Manager)" \
 userPassword
Password for user 'cn=Directory Manager':
dn: cn=Directory Manager,cn=Root DNs,cn=config
userPassword: {PBKDF2}10000:<hash>
```

Notice that the password appears neither in the shell history, nor in the terminal session.

When using scripts where the password cannot be supplied interactively, passwords can be read from files. For example, the `--bindPasswordFile` *file* option takes a file that should be readable only by the user running the command. It is also possible to set passwords in the `tools.properties` file for the user. This file is located in the user's home directory, on UNIX `~/.opendj/tools.properties`, and on Windows, typically `C:\Documents and Settings\`*username*`\.opendj\tools.properties`, though the location can depend on the Java runtime environment used. Here as well, make sure that the file is readable only by the user. Alternatively use other approaches that work with scripts such as Java properties or environment variables, depending on what method is most secure in production.

20.8.3. Passwords in Directory Data

An OpenDJ server encodes users' passwords before storing them. A variety of built-in password storage schemes are available, using either one-way (hash) or reversible algorithms. The default storage schemes use one-way algorithms to make it computationally difficult to recover the cleartext password values even when given full access to the files containing stored password values.

For details see "Configuring Password Storage".

In an OpenDJ server, password policies govern password storage schemes, valid password values, password term duration, account lockout, and others. For example you can configure password policies that prevent users from setting weak passwords and from reusing passwords. OpenDJ software provides a wide range of alternatives. For details, see "*Configuring Password Policy*".

20.9. Protect OpenDJ Server Files

By default, OpenDJ servers do not encrypt server files or directory data. The only attribute values stored in encrypted or digest form are passwords. For instructions on encrypting entries and index content, see "Encrypting Directory Data". For instructions on encrypting change log content, see "To Encrypt External Change Log Data".

If you set up an appropriate user account for the server as described in "Setting Up a System Account for an OpenDJ Server", and unpacked the server files as that user, then the system should prevent other users from having overly permissive access to server files.

Included in the files that the server does not encrypt are LDIF exports of directory data. LDIF export files are readable and writable depending on the UNIX umask or Windows file access control settings for the user who runs the command to export the LDIF. The **export-ldif** command can compress the LDIF, but does not have an option for encrypting LDIF.

Directory backup archives can be encrypted, but are not encrypted by default. Backup archive file permissions depend on the UNIX umask or Windows file access control settings. When using the **backup** command, run an online backup and supply the `--encrypt` option as shown in the following example:

```
$ backup \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword - \
 --backupAll \
 --backupDirectory /path/to/opendj/bak \
 --encrypt \
 --start 0
Password for user 'cn=Directory Manager':
Backup task <datestamp> scheduled to start...
```

The server uses its Crypto Manager configuration to determine how to encrypt the backup archive data. The `--encrypt` option is not available for offline back up. If you back up server data offline, plan to protect the files separately.

Chapter 21
Changing Server Certificates

This chapter covers how to renew certificates and replace server key pairs. In this chapter you will learn to:

• Renew an expiring certificate

• Replace a key pair for securing a connection handler

• Replace a key pair used for replication

OpenDJ server uses one keystore for key pairs to secure adminstration and client connections, and a separate keystore for replication, as shown in "Keystores and Truststores".

Keystores and Truststores

By default, keystores are found under `/path/to/opendj/config`:

* The `keystore` file holds keys for securing administrative and client connections.

    The key pair for the server has default alias `server-cert`.

    The cleartext password is stored in `keystore.pin`. It is also the private key password for `server-cert`.

    This is where you import trusted certificates for external applications.

• The `ads-truststore` file holds the server's key pair for securing replication connections, and other replicas' public key certificates.

The key pair for the server has default alias `ads-certificate`.

The cleartext password is stored in `ads-truststore.pin`. It is also the private key password for `ads-certificate`.

This keystore is synchronized with the certificates under the base DN `cn=admin data`. Do not change this keystore directly unless you understand the impact on the server configuration.

To Renew a Server Certificate

This procedure describes how to renew a certificate, for example, because the certificate is going to expire.

1. Get the certificate signed again using one of the following alternatives:

    • Self-sign the certificate.

      The following example self-signs the certificate with alias `server-cert`, setting the expiry to 7300 days (20 years) from the current time:

      ```
      $ keytool \
       -selfcert \
       -alias server-cert \
       -validity 7300 \
       -keystore /path/to/opendj/config/keystore \
       -storetype PKCS12 \
       -storepass:file /path/to/opendj/config/keystore.pin
      ```

    • Create a certificate signing request, have it signed by a CA, and import the signed certificate from the CA reply.

      For examples of the **keytool** commands to use, see "To Set Up a CA-Signed Certificate".

2. If client applications trust the self-signed certificate, have them import the renewed certificate.

    For details, see "To Import the Server Certificate".

To Replace a Server Key Pair

This procedure shows how to replace a server key pair. This procedure does not apply for replication key pairs. Instead, see "To Replace the Key Pair Used for Replication".

1. Record the alias of the existing key pair.

    This example shows a key pair that uses the default alias, `server-cert`:

```
$ keytool \
 -list \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin
...
server-cert, <date>, PrivateKeyEntry,
...
```

Connection handlers specify which key pair to use by specifying the alias in the `ssl-cert-nickname` property.

2. Generate a new key pair with a new alias.

   The following example generates a key pair that will have a self-signed certificate with alias `new-server-cert`:

```
$ keytool \
 -genkeypair \
 -keyalg RSA \
 -alias new-server-cert \
 -validity 7300 \
 -ext "san=dns:opendj.example.com" \
 -dname "CN=opendj.example.com, O=OpenDJ RSA Self-Signed Certificate" \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -keypass:file /path/to/opendj/config/keystore.pin \
 -storepass:file /path/to/opendj/config/keystore.pin
```

3. Get the new certificate signed, using one of the following alternatives:

   - Self-sign the certificate:

```
$ keytool \
 -selfcert \
 -alias new-server-cert \
 -validity 7300 \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin
```

   - Create a certificate signing request, have it signed by a CA, and import the signed certificate from the CA reply.

     For examples of the **keytool** commands to use, see "To Set Up a CA-Signed Certificate".

4. Restart the server to reload the keystore:

```
$ stop-ds --restart
```

5. For each connection handler referencing the alias of the old key pair, update the connection handler to use the new alias.

    The following example updates the LDAPS connection handler to use the new key pair:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --set listen-port:1636 \
 --set enabled:true \
 --set use-ssl:true \
 --set ssl-cert-nickname:new-server-cert \
 --trustAll \
 --no-prompt
```

    Repeat this step for all connection handlers using the old key pair.

6. If you have client applications trusting the self-signed certificate, have them import the new one.

    For details, see "To Import the Server Certificate".

7. (Optional) Once the old key pair alias is no longer used anywhere, you can optionally delete the keys using the **keytool -delete** command with the old alias.

To Replace the Key Pair Used for Replication

Follow these steps to replace the key pair that is used to secure replication connections.

1. Generate a new key pair for the server.

    The changes you perform are replicated across the topology.

    OpenDJ has an `ads-certificate` and private key, which is a local copy of the key pair used to secure replication connections.

    To generate the new key pair, you remove the `ads-certificate` key pair, prompt OpenDJ to generate a new `ads-certificate` key pair, and then add a copy to the administrative data using the MD5 fingerprint of the certificate to define the RDN.

    a. Delete the `ads-certificate` entry:

```
$ ldapmodify \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: ds-cfg-key-id=ads-certificate,cn=ads-truststore
changetype: delete

Processing DELETE request for ds-cfg-key-id=ads-certificate,cn=ads-truststore
DELETE operation successful for DN ds-cfg-key-id=ads-certificate,
 cn=ads-truststore
```

b. Prompt OpenDJ to generate a new, self-signed `ads-certificate` key pair.

    You do this by adding an `ads-certificate` entry with object class `ds-cfg-self-signed-cert-request`:

```
$ ldapmodify \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: ds-cfg-key-id=ads-certificate,cn=ads-truststore
changetype: add
objectclass: ds-cfg-self-signed-cert-request

Processing ADD request for ds-cfg-key-id=ads-certificate,cn=ads-truststore
ADD operation successful for DN ds-cfg-key-id=ads-certificate,cn=ads-truststore
```

c. Retrieve the `ads-certificate` entry:

```
$ ldapsearch \
 --port 1389 \
 --hostname opendj.example.com \
 --baseDN cn=ads-truststore \
 "(ds-cfg-key-id=ads-certificate)"
dn: ds-cfg-key-id=ads-certificate,cn=ads-truststore
ds-cfg-key-id: ads-certificate
ds-cfg-public-key-certificate;binary:: MIIB6zCCAVSgAwIBAgIEDKSUFjANBgkqhkiG9w0BA
 QUFADA6MRswGQYDVQQKExJPcGVuREogQ2VydGlmaWNhdGUxGzAZBgNVBAMTEm9wZW5hbS5leGFtcGxl
 LmNvbTAeFw0xMzAyMDcxMDMwMzNaFw0zMzAyMDIxMDMwMzNaMDoxGzAZBgNVBAoTEk9wZW5ESiBDZXJ
 0aWZpY2F0ZTEbMBkGA1UEAxMSb3BlbmFtLmV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNAD
 CBiQKBgQCfGLAiUOz4sC8CM9T5DPTk9V9ErNC8N59XwBt1aN7UjhQl4/JZZsetubtUrZBLS9cRrnYdZ
 cpFgLQNEmXifS+PdZ0DJkaLNfmd8ZX0spX8++fb4SkkggkmNRmi1fccDQ/DHMlwl7kk884lXummrzcD
 GbZ7p4vnY7y7GmD1vZSP+wIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAJciUzUP8T8A9VV6dQB0SYCNGlo
 7IvpE7jGVZh6KvM0m5sBNX3wPbTVJQNij3TDm8nx6yhi6DUkpiAZfz/OBL5k+WSw80TjpIZ2+klhP1s
 srsST4Um4fHzDZXOXHR6NM83XxZBsR6MazYecL8CiGwnYW2AeBapzbAnGn1J831q1q
objectClass: top
objectClass: ds-cfg-instance-key
```

d. Retrieve the MD5 fingerprint of the `ads-certificate`.

In this example, the MD5 fingerprint is `07:35:80:D8:F3:CE:E1:39:9C:D0:73:DB:6C:FA:CC:1C`:

```
$ keytool \
 -list \
 -v \
 -alias ads-certificate \
 -keystore /path/to/opendj/config/ads-truststore \
 -storepass:file /path/to/opendj/config/ads-truststore.pin
Alias name: ads-certificate
Creation date: Feb 7, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=opendj.example.com, O=OpenDJ Certificate
Issuer: CN=opendj.example.com, O=OpenDJ Certificate
Serial number: ca49416
Valid from: Thu Feb 07 11:30:33 CET 2013 until: Wed Feb 02 11:30:33 CET 2033
Certificate fingerprints:
  MD5:  07:35:80:D8:F3:CE:E1:39:9C:D0:73:DB:6C:FA:CC:1C
  SHA1: 56:30:F6:79:AA:C0:BD:61:88:3E:FB:38:38:9D:84:70:0B:E4:43:57
  SHA256: A8:4B:81:EE:30:2A:0C:09:2E:...:C1:41:F5:AB:19:C6:EE:AB:50:64
  Signature algorithm name: SHA1withRSA
  Version: 3
```

e. Using the MD5 fingerprint and the certificate entry, prepare LDIF to update `cn=admin data` with the new server certificate:

```
$ cat /path/to/update-server-cert.ldif
dn: ds-cfg-key-id=073580D8F3CEE1399CD073DB6CFACC1C,cn=instance keys,
 cn=admin data
changetype: add
ds-cfg-key-id: 073580D8F3CEE1399CD073DB6CFACC1C
ds-cfg-public-key-certificate;binary:: MIIB6zCCAVSgAwIBAgIEDKSUFjANBgkqhkiG9w0BA
 QUFADA6MRswGQYDVQQKExJPcGVuREogQ2VydGlmaWNhdGUxGzAZBgNVBAMTEm9wZW5hbS5leGFtcGxl
 LmNvbTAeFw0xMzAyMDcxMDMwMzNaFw0zMzAyMDIxMDMwMzNaMDoxGzAZBgNVBAoTEk9wZW5ESiBDZXJ
 0aWZpY2F0ZTEbMBkGA1UEAxMSb3BlbmFtLmV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNAD
 CBiQKBgQCfGLAiUOz4sC8CM9T5DPTk9V9ErNC8N59XwBt1aN7UjhQl4/JZZsetubtUrZBLS9cRrnYdZ
 cpFgLQNEmXifS+PdZ0DJkaLNFmd8ZX0spX8++fb4SkkggkmNRmi1fccDQ/DHMlwl7kk884lXummrzcD
 GbZ7p4vnY7y7GmD1vZSP+wIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAJciUzUP8T8A9VV6dQB0SYCNG1o
 7IvpE7jGVZh6KvM0m5sBNX3wPbTVJQNij3TDm8nx6yhi6DUkpiAZfz/OBL5k+WSw80TjpIZ2+klhPls
 srsST4Um4fHzDZXOXHR6NM83XxZBsR6MazYecL8CiGwnYW2AeBapzbAnGn1J831q1q
objectClass: top
objectClass: ds-cfg-instance-key

dn: cn=opendj.example.com:4444,cn=Servers,cn=admin data
changetype: modify
replace: ds-cfg-key-id
ds-cfg-key-id: 073580D8F3CEE1399CD073DB6CFACC1C
```

f. Update the administrative data, causing OpenDJ to create a copy of the new `ads-certificate` with its MD5 signature as the alias in the `ads-truststore`:

```
$ ldapmodify \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 /path/to/update-server-cert.ldif
Processing ADD request for ds-cfg-key-id=073580D8F3CEE1399CD073DB6CFACC1C,
 cn=instance keys,cn=admin data
ADD operation successful for DN ds-cfg-key-id=073580D8F3CEE1399CD073DB6CFACC1C,
 cn=instance keys,cn=admin data
Processing MODIFY request for cn=opendj.example.com:4444,cn=Servers,
 cn=admin data
MODIFY operation successful for DN cn=opendj.example.com:4444,cn=Servers,
 cn=admin data
```

2. Force OpenDJ to reopen replication connections using the new key pair.

   Stop replication temporarily and then start it again as described in "To Stop Replication
   Temporarily For a Replica":

```
$ dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:false \
 --no-prompt

$ dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:true \
 --no-prompt
```

Chapter 22
Moving Servers

This chapter explains how to move OpenDJ servers, which you must do if you change the hostname, the filesystem layout, or the system where the server is installed. In this chapter you will learn to:

- Prepare for the move, especially when the server is replicated, and when the directory service remains available during the move

- Perform the configuration needed to move the server

When you change where an OpenDJ server is deployed, you must take host names, port numbers, and certificates into account. The changes can also affect your replication configuration.

22.1. Overview

From time to time you might change server hardware, file system layout, or host names. At those times you move the services running on the system. You can move OpenDJ data between servers and operating systems. Most of the configuration is also portable.

Two aspects of the configuration are not portable:

1. Server certificates contain the host name of the system. Even if you did not set up secure communications when you installed the server, the server still has a certificate used for secure communications on the administrative port.

   To resolve the issue with server certificates, you can change the server certificates during the move as described in this chapter.

2. Replication configuration includes the host name and administrative port numbers.

   You can work around the issue with replication configuration by unconfiguring replication for the server before the move, and then configuring and initializing replication again after the move.

22.2. Before You Move

Take a moment to determine whether you find it quicker and easier to move your server, or to recreate a copy. To recreate a copy, install a new server, set up the new server configuration

to match the old, and then copy only the data from the old server to the new server, initializing replication from existing data, or even from LDIF if your database is not too large.

After you decide to move a server, start by taking it out of service. Taking it out of service means directing client applications elsewhere, and then preventing updates from client applications, and finally unconfiguring replication. Directing client applications elsewhere depends on your network configuration and possibly on your client application configuration. The other two steps can be completed with the **dsconfig** and **dsreplication** commands.

To Take the Server Out of Service

1. Direct client applications to other servers.

   How you do this depends on your network and client application configurations.

2. Prevent the server from accepting updates from client applications:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --hostname replica.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --set writability-mode:internal-only \
    --trustAll \
    --no-prompt
   ```

3. Unconfigure replication for the server:

   ```
   $ dsreplication \
    unconfigure \
    --unconfigureAll \
    --hostname replica.example.com \
    --port 4444 \
    --adminUID admin \
    --adminPassword password \
    --trustAll \
    --no-prompt
   ```

4. With the server no longer receiving traffic or accepting updates from clients, and no longer replicating to other servers, you can shut it down in preparation for the move:

   ```
   $ stop-ds --quiet
   ```

5. (Optional) You might also choose to remove extra log files from the server `logs/` directory before moving the server.

22.3. Moving a Server

Now that you have decided to move your server, and prepared for the move, you must not only move the files but also fix the configuration and the server certificates, and then configure replication.

To Move the Server

1. Move the contents of the server installation directory to the new location.

2. (Optional) If you must change port numbers, edit the port numbers in `config/config.ldif`, carefully avoiding changing any whitespace or other lines in the file.

3. Change server certificates as described in "*Changing Server Certificates*".

4. Start the server:

   ```
   $ start-ds --quiet
   ```

5. Configure and initialize replication:

   ```
   $ dsreplication \
    configure \
    --adminUID admin \
    --adminPassword password \
    --baseDN dc=example,dc=com \
    --host1 opendj.example.com \
    --port1 4444 \
    --bindDN1 "cn=Directory Manager" \
    --bindPassword1 password \
    --replicationPort1 8989 \
    --host2 replica.example.com \
    --port2 4444 \
    --bindDN2 "cn=Directory Manager" \
    --bindPassword2 password \
    --replicationPort2 8989 \
    --trustAll \
    --no-prompt

   $ dsreplication \
    pre-external-initialization \
    --hostname replica.example.com \
    --port 4444 \
    --adminUID admin \
    --adminPassword password \
    --baseDN dc=example,dc=com \
    --trustAll \
    --no-prompt

   $ dsreplication \
    post-external-initialization \
    --hostname replica.example.com \
    --port 4444 \
   ```

```
--adminUID admin \
--adminPassword password \
--baseDN dc=example,dc=com \
--trustAll \
--no-prompt
```

6. Accept updates from client applications:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set writability-mode:enabled \
 --trustAll \
 --no-prompt
```

7. Direct client applications to the server.

Chapter 23
Troubleshooting Server Problems

This chapter describes how to troubleshoot common server problems, and how to collect information necessary when seeking support help. In this chapter you will learn to:

- Identify server problems systematically as a first troubleshooting step

- Troubleshoot problems with installation and upgrade procedures, directory data import, data replication, and secure connections

- Reset lost administrator passwords

- Enable debug logging judiciously when solving problems

- Prevent applications from accessing the server when solving problems

- Troubleshoot problems with the way client applications access the directory

- Prepare evidence when asking a directory expert for help

23.1. Identifying the Problem

In order to solve your problem methodically, save time by defining the problem clearly up front. In a replicated environment with multiple directory servers and many client applications, it can be particularly important to pin down not only the problem (difference in observed behavior compared to expected behavior), but also the circumstances and steps that lead to the problem occurring.

Answer the following questions:

- How do you reproduce the problem?

- What exactly is the problem? In other words, what is the behavior you expected? What is the behavior you observed?

- When did the problem start occurring? Under similar circumstances, when does the problem not occur?

- Is the problem permanent? Intermittent? Is it getting worse? Getting better? Staying the same?

Pinpointing the problem can sometimes indicate where you should start looking for solutions.

23.2. Troubleshooting Installation and Upgrade

Installation and upgrade procedures result in a log file tracing the operation. The log location differs by operating system, but look for lines in the command output of the following form:

```
See file for a detailed log of this operation.
```

Prevent antivirus and intrusion detection systems from interfering with OpenDJ software.

Before using OpenDJ software with antivirus or intrusion detection software, consider the following potential problems:

Interference with normal file access

Antivirus and intrusion detection systems that perform virus scanning, sweep scanning, or deep file inspection are not compatible with OpenDJ file access, particularly database file access.

Antivirus and intrusion detection software can interfere with the normal process of opening and closing database working files. They may incorrectly mark such files as suspect to infection due to normal database processing, which involves opening and closing files in line with the database's internal logic.

Prevent antivirus and intrusion detection systems from scanning database and changelog database files.

At minimum, configure antivirus software to whitelist the OpenDJ server database files. By default, exclude the following file system directories from virus scanning:

- `/path/to/opendj/changelogDb/` (if replication is enabled)

  Prevent the antivirus software from scanning these changelog database files.

- `/path/to/opendj/db/`

  Prevent the antivirus software from scanning database files, especially `*.jdb` files.

Port blocking

Antivirus and intrusion detection software can block ports that OpenDJ uses to provide directory services.

Make sure that your software does not block the ports that OpenDJ software uses. For details, see "Limiting System and Administrative Access" in the *Security Guide*.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including OpenDJ servers.

Running antivirus software can therefore have a significant negative impact on OpenDJ server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying OpenDJ software on the same systems.

23.3. Resetting Administrator Passwords

This section describes what to do if you forgot the password for Directory Manager or for the global (replication) administrator.

Resetting the Directory Manager's Password

OpenDJ servers store the entry for Directory Manager in the LDIF representation of their configurations. You must be able to edit server files in order to reset Directory Manager's password.

1. Generate the encoded version of the new password using the OpenDJ **encode-password** command:

   ```
   $ encode-password --storageScheme SSHA512 --clearPassword password
   encode-password --storageScheme SSHA512 --clearPassword password
   ```

2. Stop the server while you edit the configuration:

   ```
   $ stop-ds --quiet
   ```

3. Find Directory Manager's entry, which has DN `cn=Directory Manager,cn=Root DNs,cn=config`, in `/path/to/opendj/config/config.ldif`, and carefully replace the `userpassword` attribute value with the encoded version of the new password, taking care not to leave any whitespace at the end of the line:

   ```
   dn: cn=Directory Manager,cn=Root DNs,cn=config
   objectClass: person
   objectClass: inetOrgPerson
   objectClass: organizationalPerson
   objectClass: ds-cfg-root-dn-user
   objectClass: top
   userpassword: <encoded-password>
   givenName: Directory
   cn: Directory Manager
   ds-cfg-alternate-bind-dn: cn=Directory Manager
   sn: Manager
   ds-pwp-password-policy-dn: cn=Root Password Policy,cn=Password Policies,cn=config
   ds-rlim-time-limit: 0
   ds-rlim-lookthrough-limit: 0
   ds-rlim-idle-time-limit: 0
   ds-rlim-size-limit: 0
   ```

4. Start the server again:

   ```
   $ start-ds --quiet
   ```

5. Verify that you can administer the server as Directory Manager using the new password:

```
$ status \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --script-friendly \
 --trustAll
Server Run Status: Started
...
```

To Reset the Global Administrator's Password

When you configure replication, part of the process involves creating a global administrator and setting that user's password. This user is present on all replicas. If you chose default values, this user has DN `cn=admin,cn=Administrators,cn=admin data`. You reset the password as you would for any other user, though you do so as Directory Manager.

1. Use the **ldappasswordmodify** command to reset the global administrator's password:

```
$ ldappasswordmodify \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --useStartTLS \
 --authzID "cn=admin,cn=Administrators,cn=admin data" \
 --newPassword password
```

2. Let replication copy the password change to other replicas.

23.4. Enabling Debug Logging

OpenDJ can write debug information and stack traces to the server debug log. What is logged depends both on debug targets that you create, and on the debug level that you choose.

To Configure Debug Logging

1. Enable the debug log, `opendj/logs/debug`, which is not enabled by default:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

2. Create a debug target or targets.

   No debug targets are enabled by default:

```
$ dsconfig \
 list-debug-targets \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --trustAll \
 --no-prompt
Debug Target : enabled : debug-exceptions-
only
-------------:---------:----------------------
```

   A debug target specifies a fully qualified OpenDJ Java package, class, or method for which to log debug messages at the level you specify:

```
$ dsconfig \
 create-debug-target \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --type generic \
 --target-name org.opends.server.api \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

3. Restart OpenDJ to see debug messages in the log:

```
$ stop-ds --restart --quiet
$ dsconfig \
 list-debug-targets \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --trustAll \
 --no-prompt
Debug Target         : enabled : debug-exceptions-
only
----------------------:---------:----------------------
org.opends.server.api : true    : false
$ tail -f /path/to/opendj/logs/debug
```

Caution

OpenDJ servers can generate a high volume of debug output. Use debug logging very sparingly on
production systems.

23.5. Preventing Access While Fixing Issues

Misconfiguration can potentially put an OpenDJ server in a state where you must intervene, and
where you need to prevent users and applications from accessing the directory until you are done
fixing the problem.

OpenDJ servers provide a *lockdown mode* that allows connections only on the loopback address, and
allows only operations requested by root DN users, such as cn=Directory Manager. You can use lockdown
mode to prevent all but administrative access while you repair a server.

To put an OpenDJ server into lockdown mode, the server must be running. You cause the server
to enter lockdown mode by using a task. Notice that the modify operation is performed over the
loopback address (accessing the OpenDJ server on the local host):

```
$ cat enter-lockdown.ldif
dn: ds-task-id=Enter Lockdown Mode,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
ds-task-id: Enter Lockdown Mode
ds-task-class-name: org.opends.server.tasks.EnterLockdownModeTask

$ ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 enter-lockdown.ldif
```

The OpenDJ server logs a notice message in `logs/errors` when lockdown mode takes effect:

```
...msg=Lockdown task Enter Lockdown Mode finished execution
```

Client applications that request operations get a message concerning lockdown mode:

```
$ ldapsearch --port 1389 --baseDN "" --searchScope base "(objectclass=*)" +
The LDAP search request failed: 53 (Unwilling to Perform)
Additional Information:  Rejecting the requested operation because the server
is in lockdown mode and will only accept requests from root users over
loopback connections
```

You also leave lockdown mode by using a task:

```
$ cat leave-lockdown.ldif
dn: ds-task-id=Enter Lockdown Mode,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
ds-task-id: Enter Lockdown Mode
ds-task-class-name: org.opends.server.tasks.EnterLockdownModeTask

$ ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 leave-lockdown.ldif
```

The OpenDJ server logs a notice message when leaving lockdown mode:

```
...msg=Leave Lockdown task Leave Lockdown Mode finished execution
```

23.6. Troubleshooting LDIF Import

By default OpenDJ requires that LDIF data you import respect standards. In particular, OpenDJ is set to check that entries to import match the schema defined for the server. You can temporarily bypass this check by using the `--skipSchemaValidation` with the **import-ldif** command.

OpenDJ also ensures by default that entries have only one structural object class. You can relax this behavior by using the advanced global configuration property, `single-structural-objectclass-behavior`. This can be useful when importing data exported from Sun Directory Server. For example, to warn when entries have more than one structural object class instead of reject such entries being added, set `single-structural-objectclass-behavior:warn` as follows:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set single-structural-objectclass-behavior:warn \
 --trustAll \
 --no-prompt
```

By default, OpenDJ also checks syntax for a number of attribute types. Relax this behavior by using the **dsconfig set-attribute-syntax-prop** command. Use the `--help` option for further information.

When running **import-ldif**, you can use the `-R` *rejectFile* option to capture entries that could not be imported, and the `--countRejects` option to return the number of rejected entries as the **import-ldif** exit code.

Once you work through the issues with your LDIF data, reinstate the default behavior to ensure automated checking.

23.7. Troubleshooting TLS/SSL Connections

This section offers tips on troubleshooting LDAPS connections and connections set up with the StartTLS operation.

In order to trust the server certificate, client applications usually compare the signature on certificates with those of the Certificate Authorities (CAs) whose certificates are distributed with the client software. For example, the Java environment is distributed with a keystore holding many CA certificates:

```
$ keytool \
 -list \
 -keystore $JAVA_HOME/jre/lib/security/cacerts \
 -storepass changeit
```

The self-signed server certificates that can be configured during OpenDJ setup are not recognized as being signed by any CAs. Your software therefore is configured not to trust the self-signed certificates by default. You must either configure the client applications to accept the self-signed certificates, or else use certificates signed by recognized CAs.

You can further debug the network traffic by collecting debug traces. To see the traffic going over TLS/SSL in debug mode, configure OpenDJ to dump debug traces from `javax.net.debug` into the `logs/server.out` file:

```
$ OPENDJ_JAVA_ARGS="-Djavax.net.debug=all" start-ds
```

23.7.1. Troubleshooting Certificates and SSL Authentication

Replication uses SSL to protect directory data on the network. In some configurations, replica can fail to connect to each other due to SSL handshake errors. This leads to error log messages such as the following:

```
...msg=SSL connection attempt from myserver (<ip>) failed:
Remote host closed connection during handshake
```

Notice these problem characteristics in the message above:

- The host name, `myserver`, is not fully qualified.

  You should not see non-fully qualified host names in the error logs. Non-fully qualified host names are a sign that an OpenDJ server has not been configured properly.

  Always install and configure OpenDJ using fully qualified host names. The OpenDJ administration connector, which is used by the **dsconfig** command, and also replication depend upon SSL and, more specifically, self-signed certificates for establishing SSL connections. If the host name used for connection establishment does not correspond to the host name stored in the SSL certificate then the SSL handshake can fail. For the purposes of establishing the SSL connection, a host name like `myserver` does not match `myserver.example.com`, and vice versa.

- The connection succeeded, but the SSL handshake failed, suggesting a problem with authentication or with the cipher or protocol negotiation. As most deployments use the same Java Virtual Machine (JVM), and the same JVM configuration for each replica, the problem is likely not related to SSL cipher or protocol negotiation, but instead lies with authentication.

Follow these steps on each OpenDJ server to check whether the problem lies with the host name configuration:

1. Make sure each OpenDJ server uses only fully qualified host names in the replication configuration. You can obtain a quick summary by running the following command against each server's configuration:

   ```
   $ grep ds-cfg-replication-server: config/config.ldif | sort | uniq
   ```

2. Make sure that the host names in OpenDJ certificates also contain fully qualified host names, and correspond to the host names found in the previous step:

```
# Examine the certificates used for the administration connector.
$ keytool -list -v -keystore config/keystore -storetype PKCS12 \
 -storepass:file config/keystore.pin | grep "^Owner:"

# Examine the certificates used for replication.
$ keytool -list -v -keystore config/ads-truststore \
 -storepass:file config/ads-truststore.pin | grep "^Owner:"
```

Sample output for a server on host `opendj.example.com` follows:

```
$ grep ds-cfg-replication-server: config/config.ldif |sort | uniq
ds-cfg-replication-server: opendj.example.com:8989
ds-cfg-replication-server: opendj.example.com:9989

$ keytool -list -v -keystore config/keystore -storetype PKCS12
 \
-storepass:file config/keystore.pin | grep "^Owner:"
Owner: CN=opendj.example.com, O=Administration Connector Self-Signed Certificate

$ keytool -list -v -keystore config/ads-truststore \
 -storepass:file config/ads-truststore.pin | grep "^Owner:"
Owner: CN=opendj.example.com, O=OpenDJ Certificate
Owner: CN=opendj.example.com, O=OpenDJ Certificate
Owner: CN=opendj.example.com, O=OpenDJ Certificate
```

Unfortunately there is no easy solution to badly configured host names. It is often easier and quicker simply to reinstall your OpenDJ servers remembering to use fully qualified host names everywhere. Consider the following:

• When using the **setup** tool to install and configure a server ensure that the `-h` option is included, and that it specifies the fully qualified host name. Make sure you include this option even if you are not enabling SSL/StartTLS LDAP connections.

   If you are using the GUI installer, then make sure you specify the fully qualified host name on the first page of the wizard.

• When using the **dsreplication** tool to enable replication make sure that any `--host` options include the fully qualified host name.

If you cannot reinstall the server, follow these steps:

1. Unconfigure replication for each replica:

```
$ dsreplication \
 unconfigure \
 --unconfigureAll \
 --port adminPort \
 --hostname hostName \
 --bindDN "cn=Directory Manager" \
 --adminPassword password \
 --trustAll \
 --no-prompt
```

2. Stop and restart each server in order to clear the in-memory ADS truststore backend.

3. Configure replication making certain that fully qualified host names are used throughout:

```
$ dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 hostName1 \
 --port1 adminPort1 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 replPort1 \
 --host2 hostName2 \
 --port2 adminPort2 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 replPort2 \
 --trustAll \
 --no-prompt
```

4. Repeat the previous step for each remaining replica. In other words, host1 with host2, host1 with host3, host1 with host4, ..., host1 with hostN.

5. Initialize all remaining replica with the data from host1:

```
$ dsreplication \
 initialize-all \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --hostname hostName1 \
 --port 4444 \
 --trustAll \
 --no-prompt
```

6. Check that the host names are correct in the configuration and in the keystores by following the steps you used to check for host name problems. The only broken host name remaining should be in the key and truststores for the administration connector:

```
$ keytool -list -v -keystore config/keystore -storetype PKCS12 \
 -storepass:file config/keystore.pin |grep "^Owner:"
```

7. Stop each server, and then fix the remaining admin connector certificate as described in "To Replace a Server Key Pair".

23.7.2. Handling Compromised Keys

As explained in "*Changing Server Certificates*", an OpenDJ server has different keys and keystores for different purposes. The public keys used for replication are also used to encrypt shared secret symmetric keys, for example, to encrypt and to sign backups. This section looks at what to do if either a key pair or secret key is compromised.

How you handle the problem depends on which key was compromised:

• For a key pair used for a client connection handler and with a certificate signed by a certificate authority (CA), contact the CA for help. The CA might choose to publish a certificate revocation list (CRL) that identifies the certificate of the compromised key pair.

  Also make sure you replace the key pair. See "To Replace a Server Key Pair" for specific steps.

• For a key pair used for a client connection handler and that has a self-signed certificate, follow the steps in "To Replace a Server Key Pair", and make sure the clients remove the compromised certificate from their truststores, updating those truststores with the new certificate.

• For a key pair that is used for replication, mark the key as compromised as described below, and replace the key pair. See "To Replace the Key Pair Used for Replication" for specific steps.

  To mark the key pair as compromised, follow these steps:

  1. Identify the key entry by searching administrative data on the server whose key was compromised.

     The server in this example is installed on opendj.example.com with administration port 4444:

```
$ ldapsearch \
 --port 1389 \
 --hostname opendj.example.com \
 --baseDN "cn=admin data" \
 "(cn=opendj.example.com:4444)" ds-cfg-key-id
dn: cn=opendj.example.com:4444,cn=Servers,cn=admin data
ds-cfg-key-id: 4F2F97979A7C05162CF64C9F73AF66ED
```

     The key ID, 4F2F97979A7C05162CF64C9F73AF66ED, is the RDN of the key entry.

  2. Mark the key as compromised by adding the attribute, ds-cfg-key-compromised-time, to the key entry.

The attribute has generalized time syntax, and so takes as its value the time at which the key was compromised expressed in generalized time. In the following example, the key pair was compromised at 8:34 AM UTC on March 21, 2017:

```
$ ldapmodify \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: ds-cfg-key-id=4F2F97979A7C05162CF64C9F73AF66ED,cn=instance keys,cn=admin data
changetype: modify
add: ds-cfg-key-compromised-time
ds-cfg-key-compromised-time: 201703210834Z
```

3. If the server uses encrypted or signed data, then the shared secret keys used for encryption or signing and associated with the compromised key pair should also be considered compromised. Therefore, mark all shared secret keys encrypted with the instance key as compromised.

To identify the shared secret keys, find the list of secret keys in the administrative data whose `ds -cfg-symmetric-key` starts with the key ID of the compromised key:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "cn=secret keys,cn=admin data" \
 "(ds-cfg-symmetric-key=4F2F97979A7C05162CF64C9F73AF66ED*)" dn
dn: ds-cfg-key-id=fba16e59-2ce1-4619-96e7-8caf33f916c8,cn=secret keys,cn=admin d
 ata

dn: ds-cfg-key-id=57bd8b8b-9cc6-4a29-b42f-fb7a9e48d713,cn=secret keys,cn=admin d
 ata

dn: ds-cfg-key-id=f05e2e6a-5c4b-44d0-b2e8-67a36d304f3a,cn=secret keys,cn=admin d
 ata
```

For each such key, mark the entry with `ds-cfg-key-compromised-time` as shown above for the instance key.

Changes to administration data are replicated to other OpenDJ servers in the replication topology.

- For a shared secret key used for data encryption that has been compromised, mark the key entry with `ds-cfg-key-compromised-time` as shown in the example above that demonstrates marking the instance key as compromised.

Again, changes to administration data are replicated to other OpenDJ servers in the replication topology.

23.8. Troubleshooting Client Operations

By default, OpenDJ servers log information about all LDAP client operations in `logs/logs/ldap-access.audit.json`. The following lines show the access log for a search operation with the filter `"(uid=bjensen)"`. The lines of JSON are formatted for readability:

```json
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": <clientPort>
  },
  "server": {
    "ip": "<clientIp>",
    "port": 1389
  },
  "request": {
    "protocol": "LDAP",
    "operation": "CONNECT",
    "connId": 0
  },
  "transactionId": "0",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 0,
    "elapsedTimeUnits": "MILLISECONDS"
  },
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
}
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": <clientPort>
  },
  "server": {
    "ip": "<clientIp>",
    "port": 1389
  },
  "request": {
    "protocol": "LDAP",
    "operation": "SEARCH",
    "connId": 0,
    "msgId": 1,
    "dn": "dc=example,dc=com",
    "scope": "sub",
    "filter": "(uid=bjensen)",
    "attrs": ["ALL"]
  },
  "transactionId": "0",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 9,
    "elapsedTimeUnits": "MILLISECONDS",
```

```
      "nentries": 1
    },
    "timestamp": "<timestamp>",
    "_id": "<uuid>"
}
{
    "eventName": "DJ-LDAP",
    "client": {
      "ip": "<clientIp>",
      "port": <clientPort>
    },
    "server": {
      "ip": "<clientIp>",
      "port": 1389
    },
    "request": {
      "protocol": "LDAP",
      "operation": "UNBIND",
      "connId": 0,
      "msgId": 2
    },
    "transactionId": "0",
    "timestamp": "<timestamp>",
    "_id": "<uuid>"
}
{
    "eventName": "DJ-LDAP",
    "client": {
      "ip": "<clientIp>",
      "port": <clientPort>
    },
    "server": {
      "ip": "<clientIp>",
      "port": 1389
    },
    "request": {
      "protocol": "LDAP",
      "operation": "DISCONNECT",
      "connId": 0
    },
    "transactionId": "0",
    "response": {
      "status": "SUCCESSFUL",
      "statusCode": "0",
      "elapsedTime": 0,
      "elapsedTimeUnits": "MILLISECONDS",
      "reason": "Client Unbind"
    },
    "timestamp": "<timestamp>",
    "_id": "<uuid>"
}
```

A message corresponds to each client connection LDAP operation. The messages include information about what operation was performed, which client requested the operation, when it was completed, and more.

When HTTP access logging is also enabled, the server does not log the internal LDAP operations corresponding to HTTP requests by default. If you want to match HTTP client operations with internal LDAP operations, prevent an OpenDJ server from suppressing internal operations by using the **dsconfig** command to set the `suppress-internal-operations` advanced property to `false` for the LDAP logger. Match the values of the `request/connId` field in the HTTP access log with the same field in the LDAP access log.

To help diagnose client errors due to access permissions, see "Viewing Effective Rights" instead.

23.8.1. Clients Need Simple Paged Results Control

For Solaris and some versions of Linux you might see a message in the OpenDJ access logs such as the following:

```
The request control with Object Identifier (OID) "1.2.840.113556.1.4.319"
cannot be used due to insufficient access rights
```

This message means clients are trying to use the simple paged results control without authenticating. By default, a global ACI allows only authenticated users to use the control.

To grant anonymous (unauthenticated) user access to the control, add a global ACI for anonymous use of the simple paged results control:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword "password" \
 --add global-aci:"(targetcontrol=\"1.2.840.113556.1.4.319\") \
 (version 3.0; acl \"Anonymous simple paged results access\"; allow(read) \
 userdn=\"ldap:///anyone\";)" \
 --trustAll \
 --no-prompt
```

Alternatively, stop the server, edit the `Anonymous control access` ACI carefully in `/path/to/opendj/config/config.ldif`, and restart the server. Unlike the **dsconfig** command, the `config.ldif` file is not a public interface, so this alternative should not be used for production servers.

23.9. Troubleshooting Replication

Replication can generally recover from conflicts and transient issues. Replication does, however, require that update operations be copied from server to server. It is therefore possible to experience temporary delays while replicas converge, especially when the write operation load is heavy. OpenDJ's tolerance for temporary divergence between replicas is what allows OpenDJ to remain available to serve client applications even when networks linking the replicas go down.

In other words, the fact that directory services are loosely convergent rather than transactional is a feature, not a bug.

That said, you may encounter errors. Replication uses its own error log file, `logs/replication`. Error messages in the log file have `category=SYNC`. The messages have the following form. Here the line is folded for readability:

```
...msg=Replication server accepted a connection from 10.10.0.10/10.10.0.10:52859
 to local address 0.0.0.0/0.0.0.0:8989 but the SSL handshake failed.
 This is probably benign, but may indicate a transient network outage
 or a misconfigured client application connecting to this replication server.
 The error was: Remote host closed connection during handshake
```

OpenDJ servers maintain historical information about changes in order to bring replicas up to date, and to resolve replication conflicts. To prevent historical information from growing without limit, servers purge historical information after a configurable delay (`replication-purge-delay`, default: 3 days). A replica can become irrevocably out of sync if you restore it from a backup archive older than the purge delay, or if you stop it for longer than the purge delay. If this happens, unconfigure the replica, and then reinitialize it from a recent backup or from a server that is up to date.

23.10. Asking For Help

When you cannot resolve a problem yourself, and want to ask for help, clearly identify the problem and how you reproduce it, and the version of the server where the problem occurs. The version includes at least a version number and a build date stamp:

```
$ status --version
OpenDJ 4.0.0
Build <datestamp>
...
```

Be prepared to provide the following additional information:

- The Java home set in `config/java.properties`.

- Access and error logs showing what the server was doing when the problem started occurring.

- A copy of the server configuration file, `config/config.ldif`, in use when the problem started occurring.

- Other relevant logs or output, such as those from client applications experiencing the problem.

- A description of the environment where the server is running, including system characteristics, host names, IP addresses, Java versions, storage characteristics, and network characteristics. This helps to understand the logs, and other information.

Index

A

Access control, 8, 64, 98
    Debugging, 122
    Disabling anonymous access, 115
    Effective rights, 122
    Evaluation, 105
    Examples, 108
    Operations, 106
    Permissions, 102
    Subjects, 103
    Targets, 100
Accounts
    Activating, 238
    Customizing notification messages, 240
    Disabling, 237
    Lockout, 235
    Status notifications, 238
Active Directory (see Pass-through authentication)
Alerts, 323

B

Backup, 30, 197, 200
    Recovery from user error, 192

C

Certificate revocation lists, 71
Certificates, 48, 70, 349, 357, 372
Commands, 14
Control panel, 11

D

Data confidentiality, 40
Database backend, 36
    Creating, 36
    Deleting, 46
    Setting disk space thresholds, 44
    Updating, 45
Debug log, 364
Directory services
    About, 1
DSML, 9, 86

E

EL expressions, 20
Encrypting data, 40, 183
Exporting data, 30
External change log, 182
    Legacy format, 189

F

File descriptors
    Requirements, 331

H

HTTP, 72

I

Importing data, 30
    Performance, 334
    Test data, 30
Indexes, 126
    Approximate, 136, 140
    Configuring, 138
    Debugging searches, 131
    Default settings, 151
    Entry limits, 147
    Equality, 135
    Extensible matching rule, 138, 140
    Ordering, 137
    Presence, 134
    Rebuilding, 145
    Substring, 136
    Verifying, 151
    Virtual list view (browsing), 138, 143

J

JMX, 88, 292
JSON, 72
JSON syntax, 141

L

LDAP
    About, 1
    Data, 2
LDAP controls
    About, 6
LDAP extended operations