Administration Guide

Mark Craig, Nemanja Lukić, Ludovic Poitou, Chris Ridd, Valery Kharseko

Table of Contents

Hands-on guide to configuring and using OpenDJ features. The OpenDJ *project offers open source LDAP directory services in Java.*

Preface

This guide shows you how to configure, maintain, and troubleshoot OpenDJ directory services. OpenDJ 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 OpenDJ administration tools
- Manage OpenDJ server processes
- Import, export, backup, and restore directory data
- Configure OpenDJ server 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 between OpenDJ directory servers
- 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 directory server resources
- Monitor directory servers through logs and alerts and over JMX
- Tune directory servers for best performance
- Secure directory server deployments
- Change directory server key pairs and public key certificates
- Move a directory server to a different system
- Troubleshoot directory server issues

Using This Guide

This guide is intended for system administrators who build, deploy, and maintain OpenDJ 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

Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server` , even if the text applies to `C:\path\to\server` as well. Absolute path names usually begin with the placeholder `/path/to/` . This path might translate to `/opt/` , `C:\Program Files\` , or somewhere else on your system. Command-line, terminal sessions are formatted as follows:

```
$ echo $JAVA_HOME
/path/to/jdk
```

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. Program listings are formatted as follows:

```
class Test {
    public static void main(String [] args)  {
        System.out.println("This is a program listing.");
    }
}
```

Accessing Documentation Online

Open Identity Platform Community publishes comprehensive documentation online:

- The Open Identity Platform Community Documentation offers a large and increasing number of up-to-date, practical articles that help you deploy and manage Open Identity Platform software.
- Open Identity Platform 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.

Joining the Open Identity Platform Community

Visit the community resource center where you can find information about each project, download nightly builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and of course get the source code as well.

Getting Support and the Contacting Open Identity Platform Community

Open Identity Platform Community Approved Vendors provide support services, professional services, trainings, and partner services to assist you in setting up and maintaining your deployments.

Understanding Directory Services

This chapter introduces directory concepts and directory 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.

OpenDJ directory services are based on the Lightweight Directory Access Protocol (LDAP). Much of this chapter serves therefore as an introduction to LDAP. OpenDJ 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.

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.

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`[1]. 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.

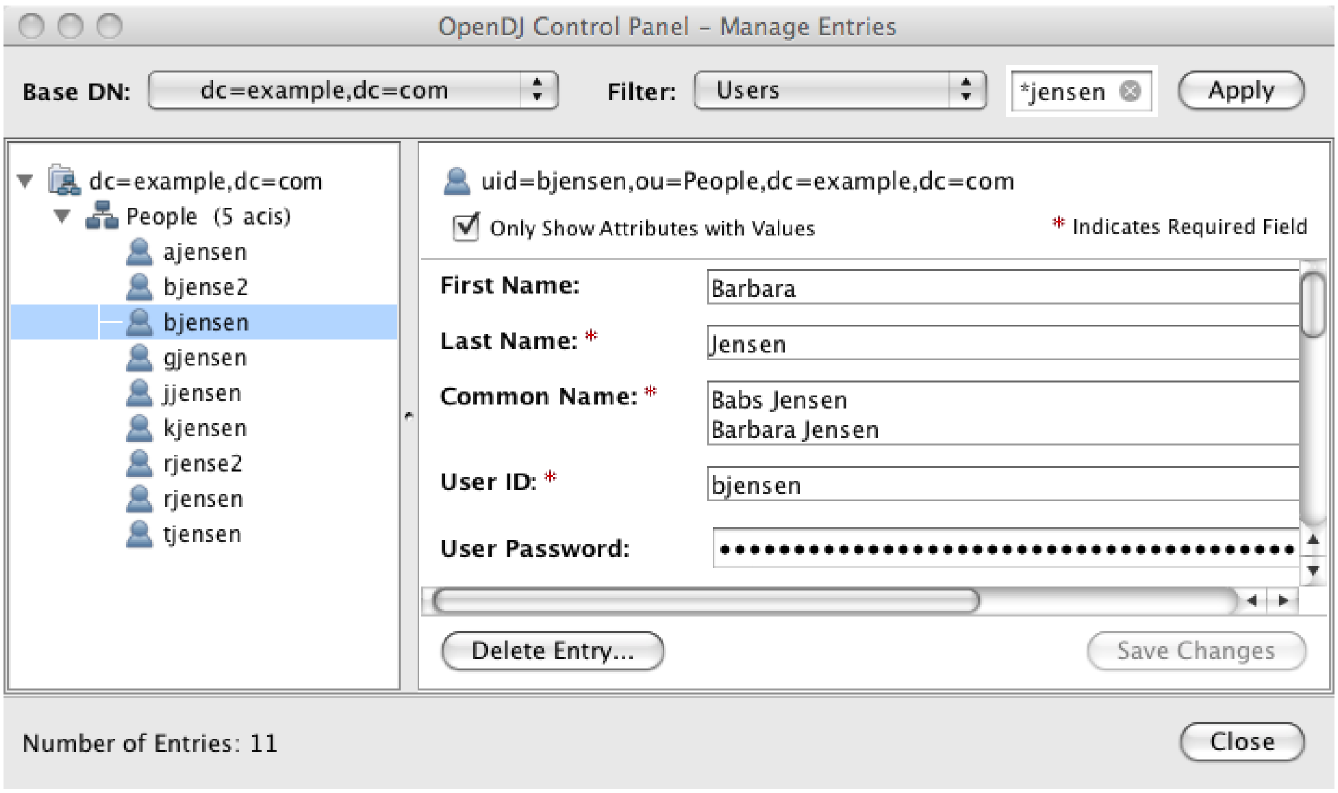
The attributes you search for are indexed in the directory, so the directory server can retrieve them more quickly.[2]

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: " # + , ; < = > \
uid: escape
cn: " # + , ; < = > \ DN Escape Characters
sn: DN Escape Characters
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, looking similar to a pyramid. [3] 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.



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.[4]

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.

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.[5] 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.
- 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. [6]
- 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 directory server, read "Performing LDAP Operations" in the *Directory Server Developer's Guide*.

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 supports 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*.

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".

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".

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".

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".

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".

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. [7] 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 your 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 also gives 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 OpenDJ itself to allow direct and more efficient HTTP and HTTPS access.

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

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.

[1] The`objectClass`attribute type indicates which types of attributes are allowed and optional 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.

[2] Attribute values do not have to be strings. Some attribute values are pure binary like certificates and photos.

[3] Hence pyramid icons are associated with directory servers.

[4] The root entry for the directory, technically the entry with DN`""`(the empty string), is called the root DSE, and contains information about what the server supports, including the other base DNs it serves.

[5] 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.

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

[7] LDAP has the object class`extensibleObject`, but its use should be the exception rather than the rule.

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

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.

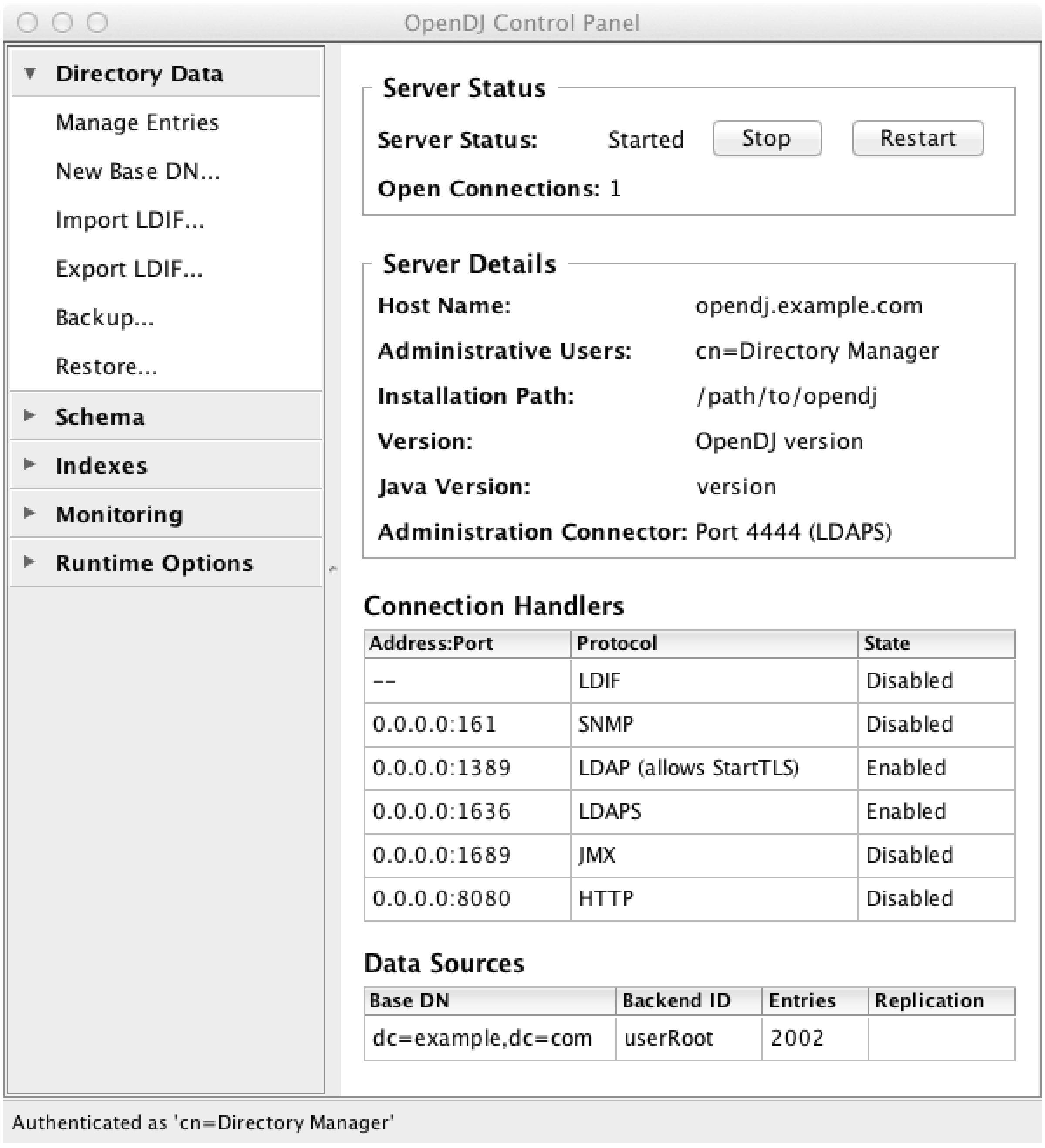
Control Panel

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 OpenDJ directory 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.



Server Status

Server Status: Started

Open Connections: 1

Server Details

Host Name:	opendj.example.com
Administrative Users:	cn=Directory Manager
Installation Path:	/path/to/opendj
Version:	OpenDJ version
Java Version:	version
Administration Connector:	Port 4444 (LDAPS)

Connection Handlers

Address:Port	Protocol	State
--	LDIF	Disabled
0.0.0.0:161	SNMP	Disabled
0.0.0.0:1389	LDAP (allows StartTLS)	Enabled
0.0.0.0:1636	LDAPS	Enabled
0.0.0.0:1689	JMX	Disabled
0.0.0.0:8080	HTTP	Disabled

Data Sources

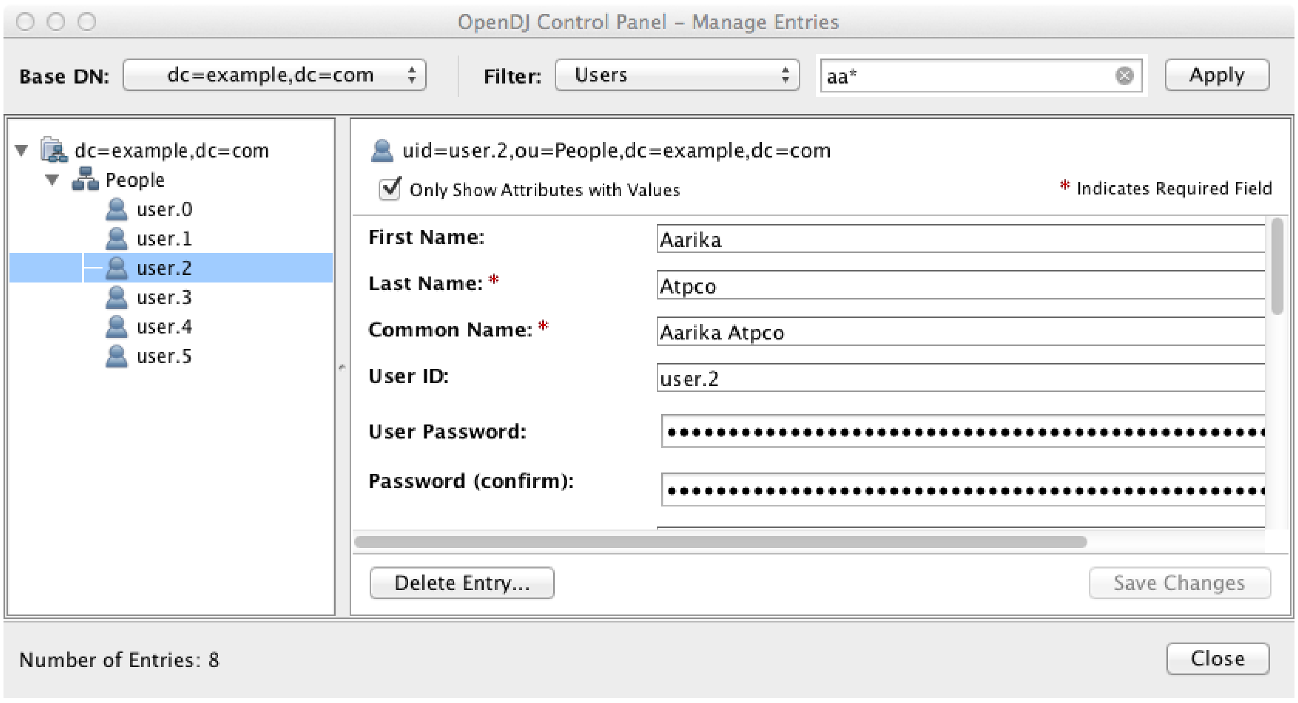
Base DN	Backend ID	Entries	Replication
dc=example,dc=com	userRoot	2002	

Authenticated as 'cn=Directory Manager'

Down the left side of 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 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.



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, preventing unindexed searches, instead adding indexes required by specific applications. (Notice that if the number of user data entries is smaller than the default resource limits, you can still perform what appear to be unindexed searches. 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 existing 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

If you did not set appropriate JVM runtime options during the installation process, this is the list that allows you to do so through the control panel.

Command-Line Tools

Before you try the examples in this guide, set your PATH to include the OpenDJ directory server tools. The location of the tools depends on the operating environment and on the packages used to install OpenDJ. "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		`/path/to/opendj/bin`
Linux distributions		`/opt/opendj/bin`
Microsoft Windows		`C:\path\to\opendj\bat`
Oracle Solaris	SVR4	`/usr/opendj/bin`

You find the installation and upgrade tools, `setup`, `upgrade`, and `uninstall`, 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 directory server.

Tools and Server Constraints

Commands	Constraints

`backendstat`
`create-rc-script`
`dsjavaproperties`
`encode-password`
`list-backends`
`setup`
`start-ds`
`upgrade`
`windows-service`
	These commands must be used with the local OpenDJ directory server in the same installation as the tools.
	`These commands are not useful with non-OpenDJ directory servers.`

Commands	Constraints

`control-panel`
`dsconfig`
`export-ldif`
`import-ldif`
`manage-account`
`manage-tasks`
`rebuild-index`
`restore`
`status`
`stop-ds`
`uninstall`
`verify-index`
	These commands must be used with OpenDJ directory 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 directory server versions. The one exception is the `dsreplication reset-change-number` subcommand, which requires OpenDJ directory server version 3.0.0 or later. `This commands is not useful with other types of directory servers.`
`make-ldif`	This command depends on template files. The template files can make use of configuration files installed with OpenDJ directory server under `config/MakeLDIF/`. `The LDIF output can be used with OpenDJ and other directory servers.`

Commands	Constraints

`base64`
`ldapcompare`
`ldapdelete`
`ldapmodify`
`ldappasswordmodify`
`ldapsearch`
`ldif-diff`
`ldifmodify`
`ldifsearch`
	These commands can be used independently of OpenDJ directory 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:

`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*.

`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`.

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.

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*.

`dsjavaproperties`

Apply changes you make to `opendj/config/java.properties`, which sets Java runtime options.

For details see dsjavaproperties(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.

Use the `ldapmodify` command with the `-a` option to add new 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*.

`ldif-diff`

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

For details see ldif-diff(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*.

`list-backends`

List backends and base DNs served by OpenDJ directory server.

For details see list-backends(1) in the *Reference*.

`make-ldif`

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

The `make-ldif` 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*.

`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*.

`start-ds`

Start OpenDJ directory 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 OpenDJ directory 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 OpenDJ as a Windows Service.

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

Managing Server Processes

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

- Start, restart, and stop OpenDJ directory server using OpenDJ command-line tools, OpenDJ control panel, or system service integration on Linux and Windows systems
- Understand how to recognize that OpenDJ directory server is recovering from a crash or abrupt shutdown
- Configure whether OpenDJ directory server loads data into cache at server startup before accepting client connections

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 OpenDJ directory 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 mark

  $ 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`.

Stopping a Server

Although OpenDJ directory server is 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:

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 \
 disable \
 --disableAll \
 --port 4444 \
 --hostname opendj.example.com \
 --adminUID admin \
 --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 OpenDJ directory 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 mark

     $ 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.

Restarting a Server

Use one of the following techniques:

- Use the `stop-ds` command:

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

- (Linux) If OpenDJ directory 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 mark

  $ /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
  ```

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:

```
111104 10:23:48:967 CONFIG [/path/to/opendj/db/userRoot]Recovery
 underway, found end of log
...
111104 10:23:49:015 CONFIG [/path/to/opendj/db/userRoot]Recovery finished:
 Recovery Info ...
```

What can take some time during server startup is preloading database content into memory when the server starts. Objects cached in memory do not survive a crash. By default, OpenDJ does not cache objects in memory before starting to accept client requests. You can, however, set the `preload-time-limit` property for the database cache of your backend if you do want to load objects into the database cache before OpenDJ begins accepting client connections.

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.

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 `make-ldif` command, described in makeldif(1) in the *Reference*.

To Generate Test LDIF Data

The `make-ldif` 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 make-ldif.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 `make-ldif` 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 `make-ldif` command to generate your LDIF file:

```
$ make-ldif \
 --randomSeed 0 \
 --templateFile /path/to/my.template \
 --ldifFile /path/to/generated.ldif
Processed 1000 entries
Processed 2000 entries
...
Processed 10000 entries
LDIF processing complete.  10003 entries written
```

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
     $ import-ldif \
      --offline \
      --includeBranch dc=example,dc=org \
      --backendID userRoot \
      --ldifFile /path/to/generated.ldif
     ```

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

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

Notice that the task is scheduled through communication over SSL on 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 default self-signed certificate used for testing.

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
$ export-ldif \
 --offline
 --includeBranch dc=example,dc=org \
 --backendID userRoot \
 --ldifFile /path/to/backup.ldif
```

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

```
$ export-ldif \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --includeBranch dc=example,dc=org \
 --backendID userRoot \
 --ldifFile /path/to/backup.ldif \
 --start 20111221230000 \
 --trustAll
```

The `--start 20111221230000` option tells OpenDJ to start the export at 11 PM on December 21, 2012.

If OpenDJ is stopped at this time, then when you start OpenDJ again, the server attempts to perform the task after starting up.

Other Tools For Working With LDIF Data

This section demonstrates the `ldifsearch`, `ldifmodify` and `ldif-diff` commands, described in ldifsearch(1) in the *Reference*, ldifmodify(1) in the *Reference*, and ldif-diff(1) in the *Reference*.

Searching in LDIF With ldifsearch

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

```
$ ldifsearch \
 --baseDN dc=example,dc=org \
 --ldifFile generated.ldif \
 "(sn=Grenier)" \
 mobile
dn: uid=user.4630,ou=People,dc=example,dc=org
mobile: +1 728 983 6669
```

The `--ldifFile ldif-file` option 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.

Updating LDIF With ldifmodify

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=org
changetype: modify
replace: description
description: This is the new description for Aaccf Amar.
-
replace: initials
initials: AAA

$ ldifmodify \
 --sourceLDIF generated.ldif \
 --changesLDIF changes.ldif \
  --targetLDIF new.ldif
```

Notice that the resulting new LDIF file is likely to be about the same size as the source LDIF file.

Comparing LDIF With ldif-diff

The `ldif-diff` command reports differences between two LDIF files in LDIF format:

```
$ ldif-diff --sourceLDIF old.ldif --targetLDIF new.ldif
dn: uid=user.0,ou=People,dc=example,dc=org
changetype: modify
add: initials
initials: AAA
-
delete: initials
initials: ASA
-
add: description
description: This is the new description for Aaccf Amar.
-
delete: description
description: This is the description for Aaccf Amar.
```

The `ldif-diff` 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.

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 directory server has tools for backing up and restoring database backends, as described in "Backing Up and Restoring Data". By default, 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 server offers a choice of JE and PDB types.

These backend types are implemented using B-tree data structures. They store data as key-value pairs, which is different from the relational model exposed to clients of relational databases. JE and PDB backends differ in how they manage data on disk:

- 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.

- A PDB backend stores data on disk using volume and journal files.

  Volume files hold the data in identically sized sections called pages. A page either holds actual data or serves as an index to other pages. If a volume file runs out of space on existing pages, the PDB backend expands the volume to add more pages. The PDB backend does not, however, shrink the volume if pages become vacant, though it can reuse free pages. Volume files stay the same size or continue to grow once you have imported the data from LDIF. Only another import operation can shrink the volume size.

  Journal files are append-only logs that record transactions and updated pages. Journal files have names like `dj_journal.number`. The PDB backend writes updates to the highest-numbered journal file. A journal file grows until it reaches 1 GB in size. The PDB backend then opens a new journal file.

  To avoid an endless increase in disk space used by journal files, PDB backends clean their journal files when idle. When the backend is idle and not in the process of being backed up, a `JOURNAL_COPIER` thread copies pages from journal files to the appropriate volume. Old journal files are deleted. If the backend is idle long enough, the PDB backend copies all updates to the volume, leaving only one small journal file.

  A PDB backend uses buffer pools in Java heap memory to cache data for fast access. Buffers are allocated to the PDB backend as long as it is in use, and are not subject to Java garbage collection. The PDB backend caches copies of data pages in the buffers, and lazily writes pages to the current journal file. At a configurable interval, the PDB backend ensures that all pages are written to disk and writes a checkpoint marker. It also writes a checkpoint marker during an orderly shutdown.

  By default, a PDB backend is configured to trade 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 before a crash. You can modify this behavior by changing the advanced configuration settings for the PDB backend.

  When a PDB backend is opened, it recovers by using its volume and journal files to recreate its B-tree structure starting with the last checkpoint marker, and then replaying more recent updates from the journal. (Recovery from an orderly shutdown is therefore optimally fast.) Recovery 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 and PDB 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*.

Creating a New Database Backend

OpenDJ stores your directory data in a *backend*. A backend is a repository that a directory server can access to store data. OpenDJ directory server offers different implementations, such as memory backends, LDIF file backends, and database backends. Database backends can be backed up and restored. By default, OpenDJ stores your data in a database backend named `userRoot`.

You can create new backends using the `dsconfig create-backend` command, described in dsconfig create-backend(1) in the *Reference*. OpenDJ directory server supports a variety of backend types, including in-memory backends, backends that store data in LDIF files, and backends that store data in key-value databases with indexes to improve performance with large data sets. When you create a backend, choose the type of backend that fits your purpose.

The following example creates a backend named `myData`. The backend is of type `pdb`, which relies on a PDB database for data storage and indexing. Alternatively, you can choose a different backend type with a different argument to the `--type` option, as in `--type je`:

```
$ dsconfig \
 create-backend \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --type pdb \
 --backend-name myData \
 --set base-dn:dc=example,dc=com \
 --set enabled:true \
 --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 myData \
 --trustAll \
 --no-prompt
Property           : Value(s)
------------------:--------------------
backend-id         : myData
base-dn            : "dc=example,dc=com"
compact-encoding   : true
db-cache-percent   : 25
db-cache-size      : 0 b
db-directory       : db
enabled            : true
index-entry-limit  : 4000
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 server creates 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".

Encrypting Directory Data

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.

NOTE This feature is new in OpenDJ directory server 3.5.

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. 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 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, 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 Directory Server Files", 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:

```
$ strings /path/to/opendj/db/userRoot/dj* | grep bjensen | sort | uniq
'uid=bjensen,ou=People,dc=example,dc=com
/home/bjensen
bjensen
bjensen@example.com
```

To maintain data confidentiality on disk, you must configure it explicitly. In addition to preventing read access by other users as described in "Set Up a System Account for OpenDJ Directory Server", you can configure confidentiality for database backends. When confidentiality is enabled for a backend, 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 "Protect Directory Server Network Connections" 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, 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. 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.

OpenDJ directory server encrypts 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", 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.

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`, OpenDJ 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`, OpenDJ server stops allowing updates, instead returning an `UNWILLING_TO_PERFORM` error to each update request.

OpenDJ server continues to apply replication updates without regard to the thresholds. OpenDJ 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
```

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).

Deleting a Database Backend

You delete a database backend by using the `dsconfig delete-backend` command, described in dsconfig delete-backend(1) in the *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".

Configuring Connection Handlers

This chapter shows you how to configure OpenDJ directory server 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

LDAP Client Access

You configure LDAP client access by using the command-line tool `dsconfig`. By default you configure OpenDJ to listen for LDAP when you install.

The standard port number for LDAP client access is 389. If you install OpenDJ directory server as a user who can use port 389 and the port is not yet in use, then 389 is the default port number presented at installation time. If you install as a user who cannot use a port < 1024, then the default port number presented at installation time is 1389.

To Change the LDAP Port Number

1. Change the port number using the `dsconfig` command:

   ```
   $ 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
   ```

   This example changes the port number to 11389 in the configuration.

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
```

Preparing For Secure Communications

One common way to protect connections between OpenDJ and client applications involves using StartTLS for LDAP or LDAPS to secure connections. OpenDJ and client applications use X.509 digital certificates to set up secure connections.

Both OpenDJ and client applications check that certificates are signed by a trusted party before accepting them. Merely setting up a secure connection therefore involves a sort of authentication using certificates. If either OpenDJ or the client application cannot trust the peer certificate, then the attempt to set up a secure connection will fail.

By default OpenDJ client tools prompt you if they do not recognize the server certificate. Other clients might not prompt you. OpenDJ server has no one to prompt when a client presents a certificate that cannot be trusted, so refuses to set up the connection.[1] In other words, it is important for both OpenDJ and client applications to be able to verify that peer certificates exchanged have been signed by a trusted party.

In practice, this means that both OpenDJ and client applications must put the certificates that were used to sign each others' certificates in their respective truststores. Conventionally, certificates are therefore signed by a Certificate Authority (CA). A CA is trusted to sign other certificates. The Java runtime environment, for example, comes with a truststore holding certificates from many well-known CAs.[2] If your client uses a valid certificate signed by one of these CAs, then OpenDJ can verify the certificate without additional configuration, because OpenDJ can find the CA certificate in the Java CA certificate truststore. Likewise, if you set up StartTLS or LDAPS in OpenDJ using a valid certificate signed by one of these CAs, then many client applications can verify the OpenDJ server certificate without further configuration.

In summary, if you need a certificate to be recognized automatically, get the certificate signed by a well-known CA.

You can, however, choose to have your certificates signed some other way. You can set up your own CA. You can use a CA whose signing certificate is not widely distributed. You can also use self-signed certificates. In each case, you must add the signing certificates into the truststore of each peer making secure connections.

For OpenDJ directory server, you can choose to import your own CA-signed certificate as part of the installation process, or later using command-line tools. Alternatively, you can let the OpenDJ installation program create a self-signed certificate as part of the OpenDJ installation process. In addition, you can add a signing certificate to the OpenDJ truststore using the Java `keytool` command.

The following example shows the `keytool` command to add a client application's binary format, self-signed certificate to the OpenDJ truststore (assuming OpenDJ is already configured to use secure connections). This enables OpenDJ to recognize the self-signed client application certificate. By definition a self-signed certificate is itself the signing certificate. Notice that the Owner and the Issuer are the same:

```
$ keytool \
 -import \
 -alias myapp-cert \
 -file myapp-cert.crt \
 -keystore /path/to/opendj/config/truststore \
 -storepass `cat /path/to/opendj/config/keystore.pin`
Owner: CN=My App, OU=Apps, DC=example, DC=com
Issuer: CN=My App, OU=Apps, DC=example, DC=com
Serial number: 5ae2277
Valid from: Fri Jan 18 18:27:09 CET 2013 until: Thu Jan 13 18:27:09 CET 2033
Certificate fingerprints:
  MD5:  48:AC:F9:13:11:E0:AB:C4:65:A2:83:9E:DB:FE:0C:37
  SHA1: F9:61:54:37:AA:C1:BC:92:45:07:64:4B:23:6C:BC:C9:CD:1D:44:0F
  SHA256: 2D:B1:58:CD:33:40:E9:ED:...:EA:C9:FF:6A:19:93:FE:E4:84:E3
  Signature algorithm name: SHA256withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 54 C0 C5 9C 73 37 85 4B   F2 3B D3 37 FD 45 0A AB  T...s7.K.;.7.E..
0010: C9 6B 32 95                                        .k2.
]
]
```

```
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

When working with a certificate in printable encoding format (.pem) rather than binary format, use the `-rfc` option, too.

Restart OpenDJ after adding certificates to the truststore to make sure that OpenDJ reads the updated truststore file.

On the client side, if your applications are Java applications, then you can also import the OpenDJ signing certificate into the trust store for the applications using the `keytool` command.

The following example shows the `keytool` command to export the OpenDJ self-signed certificate in binary format:

```
$ keytool \
 -export \
 -alias server-cert \
 -file server-cert.crt \
 -keystore /path/to/opendj/config/keystore \
 -storepass `cat /path/to/opendj/config/keystore.pin`
Certificate stored in file <server-cert.crt>
```

Importing the server certificate is similar to importing the client certificate, as shown above.

The following sections describe how to get and install certificates for OpenDJ directory server on the command-line, for use when setting up StartTLS or LDAPS.

To Request and Install a CA-Signed Certificate

First, create a server private key and public key certificate in a Java Keystore. Next, issue a signing request to the CA, and get the CA-signed certificate as a reply. Then, set up the key manager provider and trust manager provider to rely on your new server certificate stored in the OpenDJ keystore.

1. Generate the server private key and public key certificate by using the Java `keytool` command.

   The FQDN for OpenDJ directory server, which you can see 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 \
    -genkey \
    -alias server-cert \
    -keyalg rsa \
    -ext "san=dns:opendj.example.com" \
    -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
   ```

```
 -keystore /path/to/opendj/config/keystore \
 -storepass changeit \
 -keypass changeit
```

NOTE Notice that the -storepass and -keypass options take identical password arguments. OpenDJ requires that you use the same password to protect both 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 \
 -genkey \
 -alias server-cert \
 -keyalg rsa \
 -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 \
 -storepass changeit \
 -keypass changeit
```

2. Create a certificate signing request file for the certificate you generated:

```
$ keytool \
 -certreq \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass changeit \
 -file server-cert.csr
```

3. 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.

4. 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 in the reply from the (unknown) CA, keytool fails to import the signed certificate with the message keytool error: java.lang.Exception: Failed to establish chain from reply.

   The following example illustrates the import of a CA certificate created with the openssl command. See the openssl documentation for instructions on creating CAs and on signing other certificates with the CA you created:

```
$ keytool \
 -import \
 -trustcacerts \
 -keystore /path/to/opendj/config/keystore \
 -file ca.crt \
 -alias ca-cert \
 -storepass changeit
Owner: EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR
Issuer: EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR
Serial number: d4586ea05c878b0c
Valid from: Tue Jan 29 09:30:31 CET 2013 until: Mon Jan 24 09:30:31 CET 2033
Certificate fingerprints:
  MD5:  8A:83:61:9B:E7:18:A2:21:CE:92:94:96:59:68:60:FA
  SHA1: 01:99:18:38:3A:57:D7:92:7B:D6:03:8C:7B:E4:1D:37:45:0E:29:DA
  SHA256: 5D:20:F1:86:CC:CD:64:50:...:DF:15:43:07:69:44:00:FB:36:CF
  Signature algorithm name: SHA1withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 30 07 67 7D 1F 09 B6 E6   90 85 95 58 94 37 FD 31  0.g........X.7.1
0010: 03 D4 56 7B                                        ..V.
]
[EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR]
SerialNumber: [    d4586ea0 5c878b0c]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 30 07 67 7D 1F 09 B6 E6   90 85 95 58 94 37 FD 31  0.g........X.7.1
0010: 03 D4 56 7B                                        ..V.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

5. 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 `~/Downloads/server-cert.crt`:

```
$ keytool \
 -import \
 -trustcacerts \
 -alias server-cert \
 -file ~/Downloads/server-cert.crt \
 -keystore /path/to/opendj/config/keystore \
 -storepass changeit \
 -keypass changeit
Certificate reply was installed in keystore
```

6. Configure the file-based key manager provider for the Java Keystore (JKS) to use the file name and keystore PIN 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 JKS \
 --set enabled:true \
 --set key-store-pin:changeit \
 --remove key-store-pin-file:config/keystore.pin \
 --trustAll \
 --no-prompt
```

7. Configure the file-based trust manager provider.

   By convention and by default, the OpenDJ file-based trust manager provider uses a JKS file, `opendj/config/truststore`, to hold trusted public key certificates. Follow these steps to set up the truststore file, and to configure the trust manager provider.

   a. If you imported your own CA certificate into the keystore, also import the file into the truststore:

```
$ keytool \
 -import \
 -trustcacerts \
 -keystore /path/to/opendj/config/truststore \
 -file ca.crt \
 -alias ca-cert \
 -storepass changeit
Owner: EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR
Issuer: EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR
```

```
Serial number: d4586ea05c878b0c
Valid from: Tue Jan 29 09:30:31 CET 2013 until: Mon Jan 24 09:30:31 CET
2033
Certificate fingerprints:
  MD5:  8A:83:61:9B:E7:18:A2:21:CE:92:94:96:59:68:60:FA
  SHA1: 01:99:18:38:3A:57:D7:92:7B:D6:03:8C:7B:E4:1D:37:45:0E:29:DA
  SHA256: 5D:20:F1:86:CC:CD:64:50:...:DF:15:43:07:69:44:00:FB:36:CF
  Signature algorithm name: SHA1withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 30 07 67 7D 1F 09 B6 E6   90 85 95 58 94 37 FD 31  0.g........X.7.1
0010: 03 D4 56 7B                                        ..V.
]
[EMAILADDRESS=admin@example.com, CN=Example CA, O=Example Corp, C=FR]
SerialNumber: [    d4586ea0 5c878b0c]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 30 07 67 7D 1F 09 B6 E6   90 85 95 58 94 37 FD 31  0.g........X.7.1
0010: 03 D4 56 7B                                        ..V.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

b. Import the signed server certificate into the truststore:

```
$ keytool \
 -import \
 -trustcacerts \
 -alias server-cert \
 -file ~/Downloads/server-cert.crt \
 -keystore /path/to/opendj/config/truststore \
 -storepass changeit \
 -keypass changeit
```

```
Certificate was added to keystore
```

c. Configure the file-based trust manager provider to use the truststore:

```
$ dsconfig \
 set-trust-manager-provider-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name JKS \
 --set enabled:true \
 --set trust-store-file:config/truststore \
 --set trust-store-pin:changeit \
 --trustAll \
 --no-prompt
```

At this point, OpenDJ directory server can use your new CA-signed certificate, for example, for StartTLS and LDAPS connection handlers.

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 whose certificate is included with the client system, import the CA certificate into the client application truststore. Otherwise the client application cannot trust the signature on the OpenDJ CA-signed server certificate.

To Create and Install a Self-Signed Certificate

If you choose to configure LDAP secure access when setting up OpenDJ directory server, the setup program generates a key pair in the JKS `/path/to/opendj/config/keystore`, and self-signs the public key certificate, which 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 want to secure communications, but chose not to configure LDAP secure access at setup time, this procedure can help. The following steps explain how to create and install a key pair with a self-signed certificate in preparation for configuring LDAPS or HTTPS. First, create a key pair in a new JKS, and then self-sign the certificate. Next, set up the key manager provider and trust manager provider to access the new server certificate in the new 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. Generate the server certificate using the Java `keytool` command:

```
$ keytool \
 -genkey \
 -alias server-cert \
 -keyalg rsa \
 -ext "san=dns:opendj.example.com" \
 -dname "CN=opendj.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/keystore \
 -storepass changeit \
 -keypass changeit
```

In this example, OpenDJ is running on a system with fully qualified host name `opendj.example.com`. The JKS is created in the `config` directory where OpenDJ is installed, which is the default value for a JKS.

NOTE Notice that the `-storepass` and `-keypass` options take identical password arguments. OpenDJ requires that you use the same password to protect both 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 \
 -genkey \
 -alias server-cert \
 -keyalg rsa \
 -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 \
 -storepass changeit \
 -keypass changeit
```

Keep track of the password provided to the `-storepass` and `-keypass` options.

2. Self-sign the server certificate:

```
$ keytool \
 -selfcert \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass changeit
```

3. Configure the file-based key manager provider for JKS to access the Java Keystore with keystore/private key password.

   In this example, the alias is `server-cert` and the password is `changeit`.

   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:

```
$ echo changeit > /path/to/opendj/config/keystore.pin
$ chmod 600 /path/to/opendj/config/keystore.pin
$ dsconfig \
 set-key-manager-provider-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name JKS \
 --set enabled:true \
 --set key-store-file:config/keystore \
 --set key-store-pin-file:config/keystore.pin \
 --trustAll \
 --no-prompt
```

4. Configure the file-based trust manager provider for JKS to use the new server certificate.

   By convention and by default, the OpenDJ file-based trust manager provider uses a Java Keystore file, `opendj/config/truststore`, to hold trusted public key certificates. Follow these steps to set up the truststore file, and to configure the trust manager provider.

   a. Set up a truststore containing the server's public key certificate:

   ```
   $ keytool \
    -export \
    -alias server-cert \
    -keystore /path/to/opendj/config/keystore \
    -storepass changeit \
    -file server-cert.crt
   Certificate stored in file <server-cert.crt>
   $ keytool \
    -import \
    -trustcacerts \
    -alias server-cert \
    -file server-cert.crt \
    -keystore /path/to/opendj/config/truststore \
    -storepass changeit
   ...
   Trust this certificate? [no]:  yes
   Certificate was added to keystore
   ```

   b. Configure the trust manager provider to use the truststore:

   ```
   $ echo changeit > /path/to/opendj/config/truststore.pin
   $ chmod 600 /path/to/opendj/config/truststore.pin
   $ dsconfig \
    set-trust-manager-provider-prop \
    --hostname opendj.example.com \
   ```

```
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --provider-name JKS \
  --set enabled:true \
  --set trust-store-file:config/truststore \
  --set trust-store-pin-file:config/truststore.pin \
  --trustAll \
  --no-prompt
```

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

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 during installation, 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 `cat /path/to/opendj/config/keystore.pin`
server-cert, Jun 17, 2013, PrivateKeyEntry,
Certificate fingerprint (SHA1): 92:B7:4C:4F:2E:24:...:EB:7C:22:3F
```

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:JKS \
 --set trust-manager-provider:JKS \
 --trustAll \
 --no-prompt
```

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

LDAP Client Access Over SSL

You configure LDAPS (LDAP/SSL) client access by using the command-line tool `dsconfig`. You can opt to configure LDAPS access when you install.

The standard port number for LDAPS client access is 636. If you install OpenDJ directory server as a user who can use port 636 and the port is not yet in use, then 636 is the default port number presented at installation time. If you install as a user who cannot use a port < 1024, then the default port number presented at installation time is 1636.

To Set Up LDAPS Access

1. Make sure you have a server certificate installed:

```
$ keytool \
 -list \
 -alias server-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass `cat /path/to/opendj/config/keystore.pin`
server-cert, Jun 17, 2013, PrivateKeyEntry,
Certificate fingerprint (SHA1): 92:B7:4C:4F:2E:24:...:EB:7C:22:3F
```

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 listen-port:1636 \
 --set enabled:true \
 --set use-ssl:true \
 --trustAll \
 --no-prompt
```

   This example changes the port number to 1636 in the configuration.

To Change the LDAPS Port Number

1. Change the port number using the `dsconfig` command:

```
$ 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
```

This example changes the port number to 11636 in the configuration.

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
```

Restricting Client Access

Using the OpenDJ directory server global configuration properties, you can add global restrictions on how clients access the server. These settings are server-specific, and must be set independently on each server participating within the replication topology.

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 directory server supports, see "Configuring Privileges and Access Control".

Consider the following global configuration settings:

`bind-with-dn-requires-password`

Whether the directory 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set bind-with-dn-requires-password:false \
 --no-prompt
```

`max-allowed-client-connections`

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

To set a limit of 32768 use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set max-allowed-client-connections:32768 \
 --no-prompt
```

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

`return-bind-error-messages`

Does not restrict access, but by default prevents OpenDJ directory 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 OpenDJ return additional information about why a bind failed use the following command:

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

TLS Protocols and Cipher Suites

By default OpenDJ supports the SSL and TLS protocols and the cipher suites supported by the underlying Java virtual machine. For details see the documentation for the Java virtual machine (JVM) in which you run OpenDJ. For Oracle Java, see the *Java Cryptography Architecture Oracle Providers Documentation* for the The SunJSSE Provider.

To list the available protocols and cipher suites, read the `supportedTLSProtocols` and `supportedTLSCiphers` attributes of the root DSE. Install unlimited strength Java cryptography extensions for stronger ciphers:

```
$ ldapsearch --port 1389 --baseDN "" --searchScope base "(objectclass=*)" \
 supportedTLSCiphers supportedTLSProtocols
dn:
supportedTLSCiphers: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_RSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
supportedTLSCiphers: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_RSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_DHE_RSA_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_DHE_DSS_WITH_AES_128_CBC_SHA
supportedTLSCiphers: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_RSA_WITH_AES_128_GCM_SHA256
```

```
supportedTLSCiphers: TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
supportedTLSCiphers: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: SSL_RSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
supportedTLSCiphers: TLS_EMPTY_RENEGOTIATION_INFO_SCSV
supportedTLSProtocols: SSLv2Hello
supportedTLSProtocols: TLSv1
supportedTLSProtocols: TLSv1.1
supportedTLSProtocols: TLSv1.2
```

You can restrict the list of protocols and cipher suites used by setting the `ssl-protocol` and `ssl-cipher-suite` connection handler properties to include only the protocols or cipher suites you want.

For example, to restrict the cipher suites to `TLS_EMPTY_RENEGOTIATION_INFO_SCSV` and `TLS_RSA_WITH_AES_256_CBC_SHA` use the `dsconfig set-connection-handler-prop` command as shown in the following example:

```
$ dsconfig \
   set-connection-handler-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAPS Connection Handler" \
 --add ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV \
 --add ssl-cipher-suite:TLS_RSA_WITH_AES_256_CBC_SHA \
 --no-prompt \
 --trustAll
```

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. Keep in mind 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, they can use public key cryptography to authenticate each other. The server, client, or both present certificates to each other. By default, OpenDJ directory server LDAPS and HTTPS connection handlers are configured to present the server certificate, and to consider the client certificate optional. The connection handlers' `ssl-client-auth-policy` property 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 "Preparing For Secure Communications".

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, OpenDJ relies on the CA for OCSP and CRLs. This is the case even when OpenDJ is the repository for the CRLs. OpenDJ is a logical repository for certificates and CRLs. For example, 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, OpenDJ does not use the CRLs directly when checking a client certificate. Instead, when negotiating a secure connection, OpenDJ 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 OpenDJ's involvement.

After a connection is negotiated, OpenDJ directory server can authenticate a client application at the LDAP level based on the certificate. For details, see "Authenticating Using a Certificate" in the *Directory Server Developer's Guide*.

OCSP and obtaining CRLs depend on network access to the CA. If OpenDJ directory 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 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.

RESTful Client Access Over HTTP

This section describes how to use functionality in OpenDJ 3.5 and later. If you are using OpenDJ 3.0, see "RESTful Client Access (3.0)". OpenDJ offers two ways to give RESTful client applications HTTP access to directory user data as JSON resources:

- Enable the listener on OpenDJ directory 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 an HTTP Connection Handler"
  - "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 directory server also exposes 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.

OpenDJ directory server has a handler for HTTP connections. This handler exposes directory data over HTTP, including the RESTful API demonstrated in "Performing RESTful Operations" in the *Directory Server Developer's Guide*. The HTTP connection handler is not enabled by default.

Once you enable the HTTP connection handler and at least one HTTP endpoint, client applications can connect to OpenDJ directory server over HTTP. This procedure shows you how to enable the HTTP connection handler.

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.

NOTE The split between the HTTP connection handler and HTTP endpoint is new in OpenDJ 3.5.

1. Enable the connection handler:

```
$ 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 \
 --no-prompt \
 --trustAll
```

2. Enable the HTTP access log:

```
$ 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
```

This enables the HTTP access log, `opendj/logs/http-access`. For details on the format of the HTTP access log, see "Server Logs".

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

   The following example shows how to set the port to 8443, and to configure the connection handler to use transport layer security (using the default server certificate). If you did not generate a default, self-signed certificate when installing OpenDJ directory server, see "To Create and Install a Self-Signed Certificate", and more generally see "Preparing For Secure Communications" for additional instructions including how to import a CA-signed certificate:

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

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

$ stop-ds --restart
Stopping Server...
.... The Directory Server has started successfully
```

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*.

NOTE The HTTP endpoint configuration is new in OpenDJ 3.5.

The default Rest2ldap endpoint exposes the RESTful API demonstrated in "Performing RESTful Operations" in the *Directory Server 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 enabled 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 confirms the default values. 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
   {
     "_id": "bjensen",
     "_rev": "000000009ce6c3c3",
     "_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 `cat /path/to/opendj/config/keystore.pin` \
    -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
{
  "_id": "bjensen",
  "_rev": "000000009ce6c3c3",
  "_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 directory server authorizes 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.

NOTE The HTTP authentication mechanism configuration is new in OpenDJ 3.5.

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: [3]

HTTP Anonymous (enabled by default)

Handle anonymous HTTP requests, optionally binding with a specified DN.

If no bind DN is specified (default), anonymous LDAP requests are used.

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`. OpenDJ directory 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 OpenDJ directory server 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, OpenDJ directory 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), OpenDJ directory 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 OpenDJ directory 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, OpenDJ directory server requests token information from OpenAM. If the access token is valid, OpenDJ directory 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 directory 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 OpenDJ directory 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. OpenDJ directory 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, OpenDJ directory server requests token introspection from the authorization server. If the access token is valid, OpenDJ directory 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 directory server certificate if the authorization server requires mutual authentication.

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 an OpenDJ directory server configured for tests (insecure connections) to request OAuth 2.0 token information from OpenAM. Download OpenAM software from https://github.com/OpenIdentityPlatform/OpenAM/releases.

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`

Setting	Value
Rest2ldap configuration	Default settings. See "To Set Up REST Access to User Data".

Read the OpenAM 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%20wri
te" \
 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
{
  "_id": "bjensen",
  "_rev": "000000009ce6c3c3",
  "_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 directory server are under the following endpoints:

`/admin/config`

Provides a REST API to directory 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 directory 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:

     ```
     $ ldapmodify \
      --port 1389 \
      --bindDN "cn=Directory Manager" \
      --bindPassword password
     dn: uid=kvaughan,ou=People,dc=example,dc=com
     changetype: modify
     add: ds-privilege-name
     ds-privilege-name: config-read

     Processing MODIFY request for uid=kvaughan,ou=People,dc=example,dc=com
     MODIFY operation successful for DN uid=kvaughan,ou=People,dc=example,dc=com
     ```

     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:

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

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

   c. 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 directory 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 an HTTP Connection Handler".
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 `cat /path/to/opendj/config/keystore.pin` \
 -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"
{
  "_id" : "/admin",
  "_rev" : "00000000f54a6278",
  "_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"
{
  "result": [... many resources under /admin/monitor ...],
  "resultCount": 29,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

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 OpenDJ 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*.

RESTful Client Access (3.0)

NOTE This section applies to OpenDJ 3.0. For the version that applies to OpenDJ 3.5 and later, see "RESTful Client Access Over HTTP".

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

1. Enable the listener on OpenDJ directory server to respond to REST requests.

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

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

   With this approach, you must install the gateway separately.

To Set Up REST Access to OpenDJ Directory Server

OpenDJ directory server has a handler for HTTP connections where it exposes the RESTful API demonstrated in "Performing RESTful Operations" in the *Directory Server Developer's Guide*. The HTTP connection handler is not enabled by default.

You configure the mapping between JSON resources and LDAP entries by editing the configuration file for the HTTP connection handler, by default `/path/to/opendj/config/http-config.json`. The configuration is described in "REST to LDAP Configuration (3.0)" in the *Reference*. The default mapping works out of the box with Example.com data generated as part of the setup process and with Example.ldif:

1. Enable the connection handler:

```
$ 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 \
 --no-prompt \
 --trustAll
```

2. Enable the HTTP access log:

```
$ 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
```

   This enables the HTTP access log, `opendj/logs/http-access`. For details on the format of the HTTP access log, see "Server Logs".

3. (Optional) Try reading a resource.

   The HTTP connection handler paths start by default at the root context, as shown in the following example:

```
$ curl http://bjensen:hifalutin@opendj.example.com:8080/users/bjensen
{
  "_rev" : "00000000315fb731",
  "schemas" : [ "urn:scim:schemas:core:1.0" ],
  "manager" : [ {
    "_id" : "trigden",
    "displayName" : "Torrey Rigden"
  } ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "_id" : "bjensen",
  "name" : {
    "familyName" : "Jensen",
    "givenName" : "Barbara"
  },
  "userName" : "bjensen@example.com",
  "displayName" : "Barbara Jensen"
}
```

4. (Optional) If necessary, change the connection handler configuration using the `dsconfig` command.

   The following example shows how to set the port to 8443, and to configure the connection handler to use transport layer security (using the default server certificate). If you did not generate a default, self-signed certificate when installing OpenDJ directory server, see "To Create and Install a Self-Signed Certificate", and more generally see "Preparing For Secure Communications" for additional instructions including how to import a CA-signed certificate:

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

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

$ stop-ds --restart
Stopping Server...
.... The Directory Server has started successfully

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

$ curl \
 --cacert server-cert.pem \
 --user bjensen:hifalutin \
 https://opendj.example.com:8443/users/bjensen
{
  "_rev" : "0000000018c8b685",
  "schemas" : [ "urn:scim:schemas:core:1.0" ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "_id" : "bjensen",
  "name" : {
    "familyName" : "Jensen",
    "givenName" : "Barbara"
  },
  "userName" : "bjensen@example.com",
  "displayName" : "Barbara Jensen",
  "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.

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 server. Default:

localhost.

`ldap.port`

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

`ldap.userdn`

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

`ldap.userpassword`

Optional parameter specifying the password used by the DSML gateway to bind to the underlying directory server. 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 server 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`.

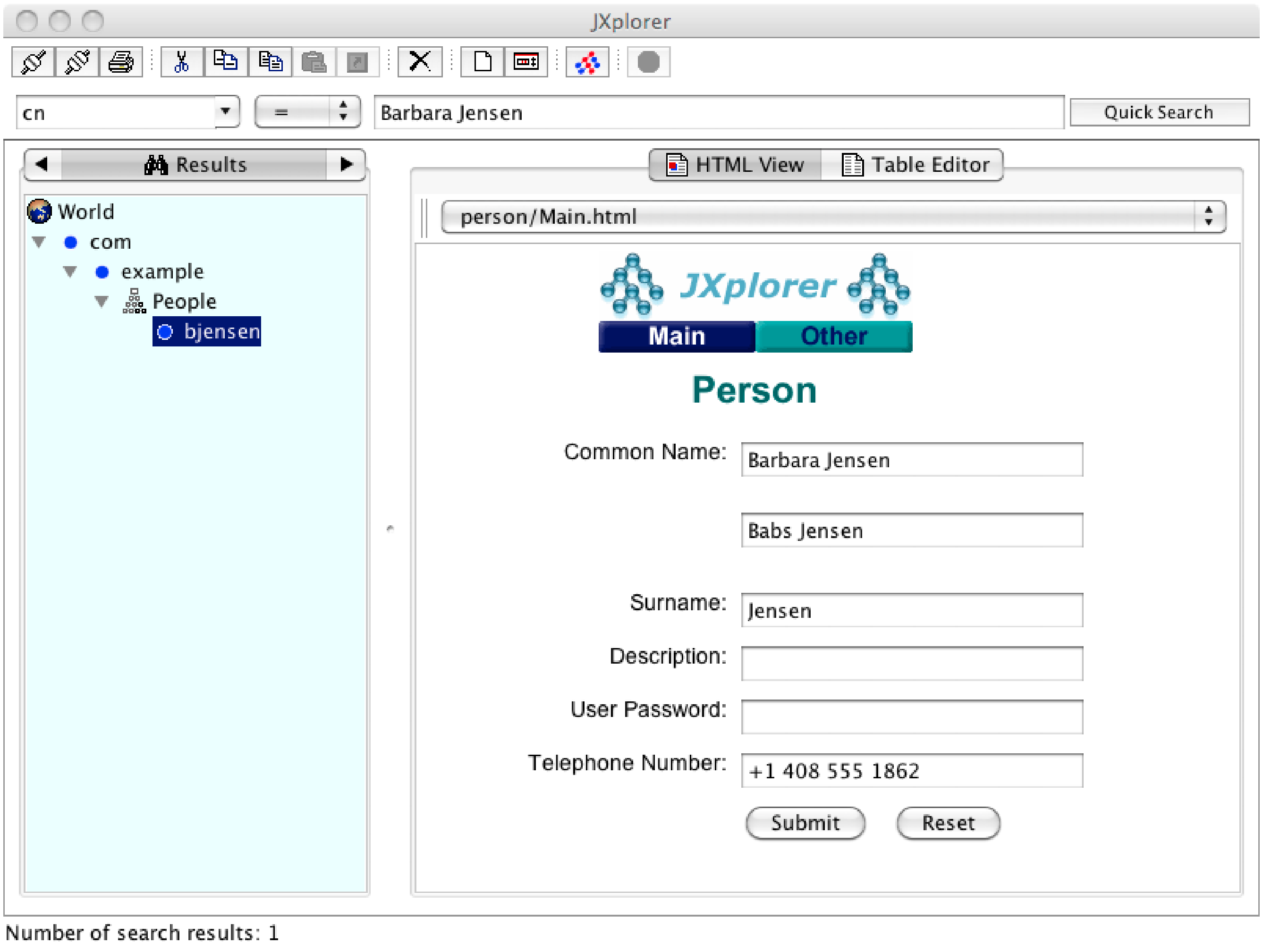
`ldap.truststore.path`

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 server. 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 OpenDJ Directory Server" shows the result.



cn = Barbara Jensen

World
com
example
People
bjensen

Person

Common Name:	Barbara Jensen
	Babs Jensen
Surname:	Jensen
Description:	
User Password:	
Telephone Number:	+1 408 555 1862

Number of search results: 1

JMX Client Access

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

To Set Up JMX Access

1. Configure the server to activate JMX access:

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

This example uses the default port number, 1689.

2. Restart the server so the change takes effect:

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

To Configure Access To JMX

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

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 OpenDJ directory server listens for connections. For example, if the server IP is `192.168.0.10` and you configured OpenDJ to listen for JMX connections on port 1689, then the service URI is `service:jmx:rmi:///jndi/rmi://192.168.0.10:1689/org.opends.server.protocols.jmx.client-unknown`.

   User name

   The full DN of the user with privileges to connect over JMX such as `uid=kvaughan,ou=People,dc=example,dc=com`.

   Password

   The bind password for the user.

LDIF File Access

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

You configure LDIF file access by using the command-line tool `dsconfig`.

To Set Up LDIF File Access

1. 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.

2. 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.

SNMP Access

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

[1] Unless you use the Blind Trust Manager Provider, which is recommended only for test purposes.

[2] `$JAVA_HOME/jre/lib/security/cacerts`holds the CA certificates. To read the full list, use the following command:

[3] The HTTP OAuth2 File mechanism is an internal interface intended for testing and not supported for production use.

Configuring Privileges and Access Control

OpenDJ supports two mechanisms to protect access to the directory, *access control instructions* and administrative *privileges*. 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`. 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.

Privileges are implemented independently from access control. By default, privileges restrict administrative access to directory root users, though any user can be assigned a privilege. Privileges apply to a directory server, and do not have a scope. This chapter covers both access control and privileges. In this chapter you will learn to:

- Configure privileges for directory administration
- 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 also access control instructions. For example, in order to reset user's passwords, an administrator needs both the `password-reset` privilege and also access control to write `userPassword` values on the user entries. By combining an access control instruction with a privilege, you can effectively restrict the scope of that privilege to a particular branch of the Directory Information Tree.

About Access Control Instructions

OpenDJ directory server access control instructions (ACIs) exist as operational `aci` attribute values on directory entries, and as global ACIs stored in the configuration. ACIs apply to a scope defined in the instruction, and 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";)
```

OpenDJ directory server's default behavior is that no access is allowed unless it is specifically granted by an access control instruction. In addition privileges assigned to certain users such as `cn=Directory Manager` allow them to bypass access control checks.

OpenDJ directory server provides several global ACIs out of the box to facilitate evaluation while maintaining a reasonable security policy. By default users are allow to read the root DSE, to read the schema, to use certain controls and extended operations, to modify their own entries, to bind, and other operations. Global ACIs are defined on the access control handler, and apply to the entire directory 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

Identify 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`).

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 also `cn=Frank Zappa,ou=Musicians,dc=example,dc=com`:

```
(target = "ldap:///*=*,*,dc=example,dc=com")
```

The *DN* must be in the subtree of the entry on which the ACI is defined.

If you do not specify `target`, then the entry holding this ACI will be affected. If `targetscope` is also omitted, then this entry and all subordinates will be 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 needed. You can use a plus sign, +, to specify all operational attributes.

Note that 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.

Here *expression* takes one of the following forms. Separate expressions with semicolons (;):

```
op=attr1:filter1[&& attr2:filter2 …][;op=attr3:filter3[&& attr4:filter4 …] …]
```

Here *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")`

Here `base` refers to the entry where the ACI is defined, `onelevel` to immediate children, `subtree` to the base entry and all children, and `subordinate` to all children only.

If you do not specify `targetscope`, then 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.

NOTE

Different LDAP server implementations that support Netscape's ACI syntax may support different multi-valued quotation styles or policies. Specifically, this can relate to `attr-list` and `OID` values.

OpenDJ ONLY offers support for the so-called "All-Encompassing" quotation style, as is demonstrated throughout this guide. For instance:

`(targetattr = "attr1 || attr2 || attr3")`

Other implementations may also support the so-called "Individual" quotation style, which is expressed as: `(targetattr = "attr1" || "attr2" || "attr3")`

Users migrating to OpenDJ from an implementation that not only supports the "Individual" quotation style, but is actively using it, will need to take care to sanitize any inbound ACIs bearing this style of quotation, else errors will occur during integration.

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 where needed. What looks harmless and simple in your lab 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. Needs to be combine with `read` in order to read the search results.

`selfwrite`
: Add or delete own DN from a group.

`write`
: Modify attributes on ACI target entries.

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 subject condition should match the value or `!=` to specify that the subject condition should not match the value:

`authmethod [!]= "none|simple|ssl|sasl mech"`
: Here you use `none` to mean do not check, `simple` for simple authentication, `ssl` for certificate-based authentication over LDAPS, `sasl mech` for 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"`

You can use asterisks, , **to replace name components, such as `dns = ".myCompany.com"`**.

`groupdn [!]= "ldap:///DN[|| ldap:///DN …]"`

Replace *DN* with the distinguished name of a group to permit or restrict access for members.

`ip [!]= "addresses"`

Here *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, and 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"`

Here the security strength factor pertains to the cipher key strength for connections using DIGEST-MD5, GSSAPI, SSL, or TLS. For example, to require that the connection must have at least 128 bits of encryption, specify `ssf >= "128"`.

`timeofday = "hhmm"`,`timeofday != "hhmm"`,`timeofday > "hhmm"`,`timeofday >= "hhmm"`,`timeofday < "hhmm"`,`timeofday ⇐ "hhmm"`

Here *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. To get the attributes of the bind entry, OpenDJ performs an internal search for the user attributes. This ACI subject therefore 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. Here *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`, `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.

How ACI is Evaluated

Understanding how OpenDJ evaluates the `aci` values is critical when implementing an access control policy. The rules the server follows are simple:

1. To determine if an operation is allowed or denied, the 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 matches the target and denies the required access, and the other list contains all the aci values that matches the target and allows the required access.
3. If the deny list contains any aci values after this procedure, access will be immediately denied.
4. If the deny list is empty, then the allow list is processed. If the allow list contains any aci values, access will be allowed.
5. If both lists are empty, access will be 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.

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 will automatically allow 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");)
```

About Privileges

Privileges provide access control for server administration independently from access control instructions.

Directory root users, such as cn=Directory Manager, are granted privileges in the following list and marked with an asterisk (*) by default. Other administrator users can be assigned 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

`proxied-auth`

Use the Proxied Authorization control

`server-lockdown`*

Put OpenDJ into and take OpenDJ 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 correponding index

`update-schema`*

Change OpenDJ schema definitions

- = default directory root user privileges

Configuring Privileges

For root directory administrators, by default `cn=Directory Manager`, you configure privileges using the `dsconfig` command.

For non-root directory administrators, you add privileges with the `ldapmodify` command.

To Change Root DN Privileges

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 on an Individual Entry

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

1. Determine the privileges to add:

   ```
   $ cat privilege.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 reset a user password:

   ```
   $ ldapsearch \
    --port 1389 \
    --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
    --bindPassword bribery \
    --baseDN cn=config \
    "(objectclass=*)"
   SEARCH operation failed
   Result Code:  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 with result code 50
Error Message:  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 \
 --filename privilege.ldif
Processing MODIFY request for uid=kvaughan,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=kvaughan,ou=People,dc=example,dc=com
```

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
ds-cfg-return-bind-error-messages: false
ds-cfg-default-password-policy: cn=Default Password Policy,cn=Password
Policies,
 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, you no doubt use a group to define adminstrative 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. OpenDJ extends collective attributes to give you fine-grained control over the which entries in the subtree are targeted.

Also, by also extending the RFC 3672 `SpecificationFilter` component, users may leverage virtual attributes, such as `isMemberOf`, to construct a search filter for targeting entries to which the collective attributes apply. This allows you, for example, to define administrative privileges that apply to all users who belong to an administrator group.

In addition to this feature, the traditional `Refinement ASN.1 CHOICE component` — also defined within RFC 3672 — is supported for use as a `SpecificationFilter` statement as well.

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 \
 --defaultAdd \
 --filename collective.ldif
Processing ADD request for cn=Administrator Privileges,dc=example,dc=com
ADD operation successful for DN cn=Administrator Privileges,dc=example,dc=com
```

   The Directory Administrators group for Example.com includes members like Kirsten Vaughan.

2. Observe that the change takes effect immediately:

```
$ 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 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. OpenDJ also assigns default directory root user privileges. In some cases the list of inherited privileges can be too broad. OpenDJ has a mechanism to limit the privileges assigned 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 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: config-write
ds-privilege-name: ldif-export
ds-privilege-name: modify-acl
ds-privilege-name: password-reset
ds-privilege-name: proxied-auth
```

2. Set the privilege attribute for the account to deny the privilege:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: uid=kvaughan,ou=people,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: -password-reset

Processing MODIFY request for uid=kvaughan,ou=people,dc=example,dc=com
MODIFY operation successful for DN uid=kvaughan,ou=people,dc=example,dc=com
```

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" \
 --newPassword changeit
The LDAP password modify operation failed with result code 50
Error Message:  You do not have sufficient privileges to perform password
reset operations
```

Configuring Access Control

Access control instructions are defined in the data as values for `aci` attributes. They can be imported in LDIF and modified over LDAP. Yet in order to make changes to ACIs users first need the `modify-acl` privilege described previously. 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`. For a list, see "Default Global ACIs". You can modify global ACIs from the Access Control Handler menu in `dsconfig`. 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.

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";)`

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 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";)`
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";)`

Name	Description	ACI Definition
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 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";)`

Name	Description	ACI Definition
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";)`
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";)`
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";)`
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";)`

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 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";)`
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 therefore 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 your syntax and lets you know if you made an error before it saves any changes.

For hints on updating directory entries with the `ldapmodify` command, see "Modifying Entry Attributes" in the *Directory Server Developer's Guide*, keeping in mind that the name of the ACI attribute is `aci` as shown in the examples that follow.

ACI: Anonymous Reads and Searches

This works when the only attributes you do not want world-readable are 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 OpenDJ denies access unless an access control explicitly allows access.[1] However, OpenDJ also allows anonymous access by default to use some controls, to perform certain extended operations, to view root DSE operational attributes, to view directory schema definitions, to view some other operational attributes, and to perform compare and search operations.

These default capabilities are defined on the `global-aci` property of the access control handler, which you can read by using the `dsconfig get-access-control-handler-prop` command:

```
$ dsconfig \
 get-access-control-handler-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --property global-aci
```

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 OpenDJ directory 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 directory 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 some controls and extended operations:

```
$ dsconfig \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --commandFilePath /tmp/captured-global-aci-edits.sh

# The dsconfig command runs interactively.

# 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 first remove the existing values,
# then add the edited values, and finally apply the changes.
```

Make sure that you also set appropriate ACIs on any data that you import.

At this point, clients must authenticate to view search results, for example:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)"
$ ldapsearch  \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin  \
 --port 1389 \
 --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
```

An example of the captured command is the shell script, captured-global-aci-edits.sh.

To reject anonymous access except bind and StartTLS requests, set `reject-unauthenticated-requests:true`:

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

Once you set the property, anonymous clients trying to search, for example, get an `Unwilling to Perform` response from OpenDJ directory server:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=bjensen)"
SEARCH operation failed
Result Code:  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 directory server configuration, and so are not replicated to other servers. You must instead apply the changes separately to each server.

ACI: Full Access for Administrators

Directory Administrators need privileges as well for full access to administrative operations:

```
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 data import and export operations.

ACI: Change Your Password

By default this capability is set in a global ACI:

```
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:

```
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

Let 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.

Viewing Effective Rights

Once you set up a number of ACIs, you might find it difficult to understand by inspection what rights a user actually has to a given entry. The Get Effective Rights control can help.

NOTE The control OID, `1.3.6.1.4.1.42.2.27.9.5.2`, is not allowed by the default global ACIs.

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=*"
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
objectClass: groupOfNames
objectClass: top
member: uid=bjensen,ou=People,dc=example,dc=com
description: People who are willing to carpool
owner: uid=bjensen,ou=People,dc=example,dc=com
cn: Carpoolers
```

Performing the same search with the get effective rights control, and asking for the `aclRights` attribute, shows what rights Babs has on the entry:

```
$ ldapsearch \
 --control effectiverights \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "cn=*" \
 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
```

When you request the `aclRightsInfo` attribute, the server responds with information about the ACIs applied:

```
$ ldapsearch \
 --control effectiverights \
 --port 1389 \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "cn=*" \
 aclRights \
 aclRightsInfo
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
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 (uid=bjensen,ou=People,dc=example,dc=com) (not proxied) ( reason: evaluat
 ed allow , deciding_aci: Anonymous read-search access)
aclRightsInfo;logs;entryLevel;write: acl_summary(main): access not allowed(write
 ) on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL
 ) to (uid=bjensen,ou=People,dc=example,dc=com) (not proxied) ( reason: no acis
 matched the subject )
aclRightsInfo;logs;entryLevel;add: acl_summary(main): access not allowed(add) on
  entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL) to
  (uid=bjensen,ou=People,dc=example,dc=com) (not proxied) ( reason: no acis matc
 hed the subject )
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 (uid=bjensen,ou=People,dc=example,dc=com) (not proxied) ( reason: evaluated
 allow , deciding_aci: Owner can delete self service groups)
aclRights;entryLevel: add:0,delete:1,read:1,write:0,proxy:0
aclRightsInfo;logs;entryLevel;proxy: acl_summary(main): access not allowed(proxy
 ) on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL
 ) to (uid=bjensen,ou=People,dc=example,dc=com) (not proxied) ( reason: no acis
 matched the subject )
```

You can also request the effective rights for another user by using the `--getEffectiveRightsAuthzid` (short form: `-g`) option, which takes the authorization identity of the other user as an argument. The following example shows Directory Manager checking 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=Directory Manager" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "cn=*" aclRightsInfo
dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
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 (anonymous) (not proxied) ( reason: evaluated allow , deciding_aci: Anony
 mous read-search access)
aclRightsInfo;logs;entryLevel;write: acl_summary(main): access not allowed(write
 ) on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL
 ) to (anonymous) (not proxied) ( reason: no acis matched the subject )
aclRightsInfo;logs;entryLevel;add: acl_summary(main): access not allowed(add) on
  entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL) to
  (anonymous) (not proxied) ( reason: no acis matched the subject )
aclRightsInfo;logs;entryLevel;delete: acl_summary(main): access not allowed(dele
 te) on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NU
 LL) to (anonymous) (not proxied) ( reason: no acis matched the subject )
aclRightsInfo;logs;entryLevel;proxy: acl_summary(main): access not allowed(proxy
 ) on entry/attr(cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com, NULL
 ) to (anonymous) (not proxied) ( reason: no acis matched the subject )
```

When you need to check access to an attribute that might not yet exist on the entry, use the `--getEffectiveRightsAttribute` (short form: `-e`) option, which takes an attribute list as an argument. The following example shows Directory Manager checking anonymous user access to the description attribute for the Self Service groups organizational unit entry. The description attribute is not yet in the entry:

```
$ ldapsearch \
 --port 1389 \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "ou=Self Service" description
dn: ou=Self Service,ou=Groups,dc=example,dc=com

$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:" \
 --getEffectiveRightsAttribute description \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "ou=Self Service" aclRights
dn: ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;attributeLevel;description: search:1,read:1,compare:1,write:0,selfwrit
 e_add:0,selfwrite_delete:0,proxy:0
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

[1] This does not apply to the directory root user, such as`cn=Directory Manager`, who bypasses ACIs.

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

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.

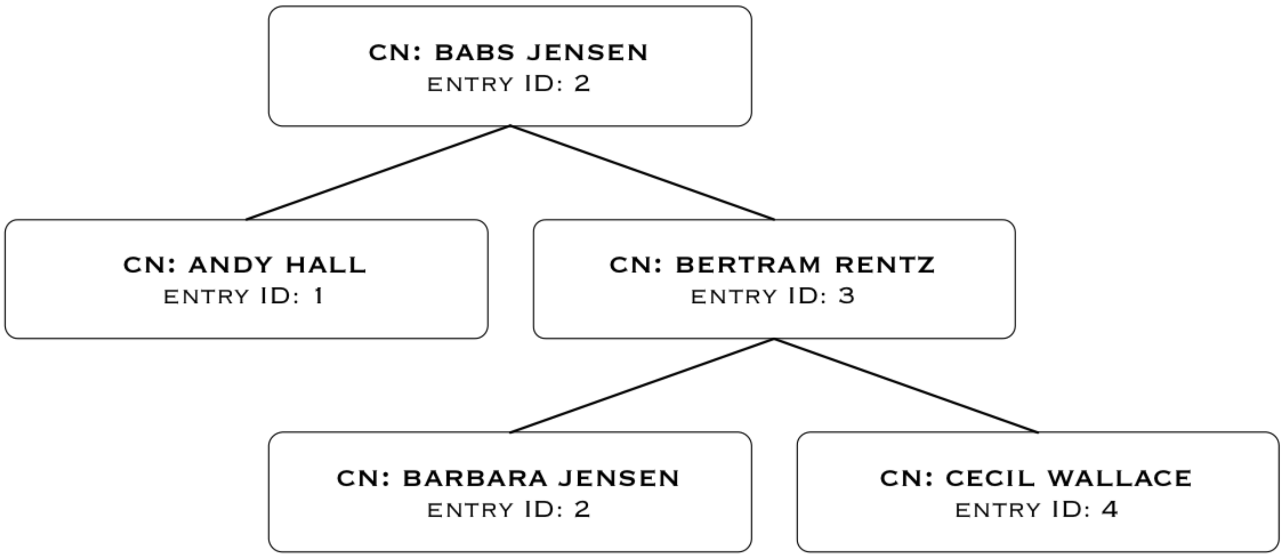
  A search filter, such as `(cn=Babs Jensen)`, asserts that an attribute on the entry to search for, in this case `cn`, corresponds to some value. In this case, the attribute must have a value that equals `Babs Jensen`, ignoring case sensitivity.

  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.

Directories 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, 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.



This is how OpenDJ directory server uses indexes. When the search filter is `(cn=Babs Jensen)`, 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, 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, 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, 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 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, 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.

What To Index

OpenDJ search performance depends on indexes as described in "About Indexes".

OpenDJ directory server maintains generally useful indexes for data imported into the default `userRoot` backend. When you create a new backend, 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 directory server, 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".

Determining Which Indexes Are Needed

Index maintenance has its costs. Every time an indexed attribute is updated, 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 OpenDJ directory server to operate properly. The latter indexes include non-configurable internal indexes, and generally are handled by 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 `unindexed`:

```
$ grep -B 1 unindexed /path/to/opendj/logs/access
SEARCH REQ conn=5 op=0 msgID=1 base="ou=people,dc=example,dc=com" scope=sub
 filter="(&(mail=*.net)(objectclass=person))" attrs="ALL"
SEARCH RES conn=5 op=0 msgID=1 result=50 message="You do not have sufficient
 privileges to perform an unindexed search" nentries=0 unindexed etime=9
--
SEARCH REQ conn=9 op=0 msgID=1 base="ou=people,dc=example,dc=com" scope=sub
 filter="(&(employeenumber=86182)(mail=*@maildomain.net))" attrs="ALL"
SEARCH RES conn=9 op=0 msgID=1 result=50 message="You do not have sufficient
 privileges to perform an unindexed search" nentries=0 unindexed etime=3
--
SEARCH REQ conn=11 op=0 msgID=1 base="ou=people,dc=example,dc=com" scope=sub
 filter="(objectclass=person)" attrs="ALL"
SEARCH RES conn=11 op=0 msgID=1 result=50 message="You do not have sufficient
 privileges to perform an unindexed search" nentries=0 unindexed etime=3
```

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 directory server provides 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set index-filter-analyzer-enabled:true \
 --no-prompt \
 --trustAll
```

The command causes OpenDJ directory 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 \
 --baseDN "cn=userRoot Storage,cn=monitor" \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 "(objectclass=*)" \
 filter-use
dn: cn=userRoot Storage,cn=monitor
filter-use: (objectClass=ldapSubentry) hits:1 maxmatches:0 message:
filter-use: (aci=*) hits:1 maxmatches:0 message:
filter-use: (employeenumber=86182) hits:6 maxmatches:-1 message:equality index t
 ype is disabled for the employeeNumber attribute
filter-use: (mail=*@maildomain.net) hits:6 maxmatches:-1 message:The filter valu
 e exceeded the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5S
 ubstringsMatch:6 index...
filter-use: (objectClass=subentry) hits:1 maxmatches:0 message:
filter-use: (cn=aa*) hits:2 maxmatches:50 message:
filter-use: (objectClass=ds-virtual-static-group) hits:1 maxmatches:0 message:
filter-use: (objectClass=groupOfNames) hits:1 maxmatches:0 message:
filter-use: (uid=user.86182) hits:2 maxmatches:1 message:
filter-use: (mail=*.net) hits:1 maxmatches:-1 message:The filter value exceeded
 the index entry limit for the /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMa
 tch:6 index
filter-use: (objectclass=person) hits:3 maxmatches:-1 message:The filter value e
 xceeded the index entry limit for the /dc=com,dc=example/objectClass.objectIden
 tifierMatch index
filter-use: (objectClass=groupOfEntries) hits:1 maxmatches:0 message:
filter-use: (objectClass=groupOfUniqueNames) hits:1 maxmatches:0 message:
filter-use: (objectClass=groupOfURLs) hits:1 maxmatches:0 message:
```

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.

Notice in the example output above that you see filters for internal use, such as `(aci=*)`. You also see filters for searches that are not indexed.

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=*.net)`, 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 `.net`. There are so many such entries that although an index exists for the `mail` attribute, OpenDJ directory server has given up maintaining the list of entries with email addresses ending in `.net`. 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 OpenDJ directory 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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. 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 `.net`:

```
$ ldapsearch \
 --port 1389 \
 --baseDN ou=people,dc=example,dc=com \
 "(&(mail=*.net)(objectclass=person))"
SEARCH operation failed
Result Code:  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".

Clarifying Which Indexes Are Used by a Search

Sometimes it is not obvious by inspection how OpenDJ directory server handles a given search request internally. The directory root user can inspect how OpenDJ directory server resolves the search request by performing the same search with the `debugsearchindex` attribute. The following example demonstrates this feature for an exact match search:

```
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 "(uid=user.1000)" \
 debugsearchindex
dn: cn=debugsearch
debugsearchindex: filter=(uid=user.1000)[INDEX:uid.equality][COUNT:1] final=[COU
 NT:1]
```

When you request the `debugsearchindex` attribute, instead of performing the search, OpenDJ directory server returns debug information indicating how it would process the search operation. In the example above, notice that 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 OpenDJ directory server would have to evaluate over 10,000 entries:

```
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 "(uid=*)" \
 debugsearchindex
dn: cn=debugsearch
debugsearchindex: filter=(uid=*)[NOT-INDEXED] scope=sub[LIMIT-EXCEEDED:10002]
 final=[NOT-INDEXED]
```

Although an index exists, the set of results is so large that OpenDJ directory 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 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-idscope-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.

Index Types and Their Functions

OpenDJ directory server supports 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.

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=*)" -
dn: dc=example,dc=com

dn: ou=People,dc=example,dc=com
```

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 directory server index is implemented as a tree of key-value pairs. The following command examines the ACI presence index for Example.ldif data:

```
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName aci.presence
Key (len 1): PRESENCE
Value (len 5): [COUNT:2] 100003 100011

Total Records: 1
Total / Average Key Size: 1 bytes / 1 bytes
Total / Average Data Size: 5 bytes / 5 bytes
```

In this case, there are two entries that have ACI attributes. Their IDs are `100003` and `100011`.

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 directory server index is implemented as a tree of key-value pairs. The following command examines the SN equality index for Example.ldif data:

```
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreMatch
...
Key (len 6): jensen
Value (len 12): [COUNT:9] 100018 100031 100032 100066 100079 100094 100133
 100134 100150
...

Total Records: 87
Total / Average Key Size: 528 bytes / 6 bytes
Total / Average Data Size: 414 bytes / 4 bytes
```

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, OpenDJ directory server can reuse an equality index for some other searches, such as ordering and initial substring searches.

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 directory server index is implemented as a tree of key-value pairs. The following command examines an SN approximate index for Example.ldif data:

```
$ 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] 100018 100031 100032 100059 100066 100079 100094
 100133 100134 100150
...

Total Records: 84
Total / Average Key Size: 276 bytes / 3 bytes
Total / Average Data Size: 405 bytes / 4 bytes
```

In this case, there are ten entries that have an SN that sounds like Jensen.

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 OpenDJ directory 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 directory server index is implemented as a tree of key-value pairs. The following command examines an SN substring index for Example.ldif data:

```
$ backendstat \
 dump-index \
 --backendID userRoot \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreSubstringsMatch:6
...
Key (len 1): e
Value (len 25): [COUNT:22] 100024 100027 100035 100046 100048 100052 100058
 100070 100073 100074 100075 100080 100091 100093 100100 100115 100117 100123
 100142 100148 100152 100155
...
Key (len 2): en
Value (len 15): [COUNT:12] 100018 100031 100032 100037 100066 100079 100094
 100122 100133 100134 100150 100156
...
Key (len 3): ens
Value (len 4): [COUNT:1] 100147
Key (len 5): ensen
Value (len 12): [COUNT:9] 100018 100031 100032 100066 100079 100094 100133
 100134 100150
...
Key (len 6): jensen
Value (len 12): [COUNT:9] 100018 100031 100032 100066 100079 100094 100133
 100134 100150
...
Key (len 1): n
Value (len 35): [COUNT:32] 100013 100014 100018 100019 100020 100022 100031
 100032 100037 100049 100054 100059 100066 100071 100077 100079 100088 100094
 100097 100102 100106 100113 100116 100122 100124 100133 100134 100143 100144
 100150 100153 100156
...
Key (len 2): ns
```

```
Value (len 4): [COUNT:1] 100147
Key (len 4): nsen
Value (len 12): [COUNT:9] 100018 100031 100032 100066 100079 100094 100133
 100134 100150
...
Key (len 1): s
Value (len 15): [COUNT:12] 100012 100026 100047 100064 100095 100098 100108
 100131 100135 100147 100149 100154
...
Key (len 2): se
Value (len 9): [COUNT:6] 100052 100058 100075 100117 100123 100148
Key (len 3): sen
Value (len 12): [COUNT:9] 100018 100031 100032 100066 100079 100094 100133
 100134 100150
...

Total Records: 391
Total / Average Key Size: 1653 bytes / 4 bytes
Total / Average Data Size: 2095 bytes / 5 bytes
```

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.

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 OpenDJ directory 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, OpenDJ directory 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.

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.

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.

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"

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".

Create a New Index

The following example creates a new equality index for the `cn` (common name) attribute in a backend of type `pdb` named `myData`:

```
$ dsconfig \
 create-backend-index \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name myData \
 --index-name cn \
 --set index-type:equality \
 --trustAll \
 --no-prompt
```

Configure an Approximate Index

The following example configures an approximate index for the `cn` (common name) attribute in a backend of type `pdb` named `myData`:

```
$ dsconfig \
 set-backend-index-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name myData \
 --index-name cn \
 --set 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 directory server supports matching rules defined in LDAP RFCs. It also defines 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 "Configure an Approximate Index".

For an example using this matching rule, see "Search: Finding an Approximate Match" in the *Directory Server 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 *Directory Server 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 *Directory Server 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 *Directory Server Developer's Guide*.

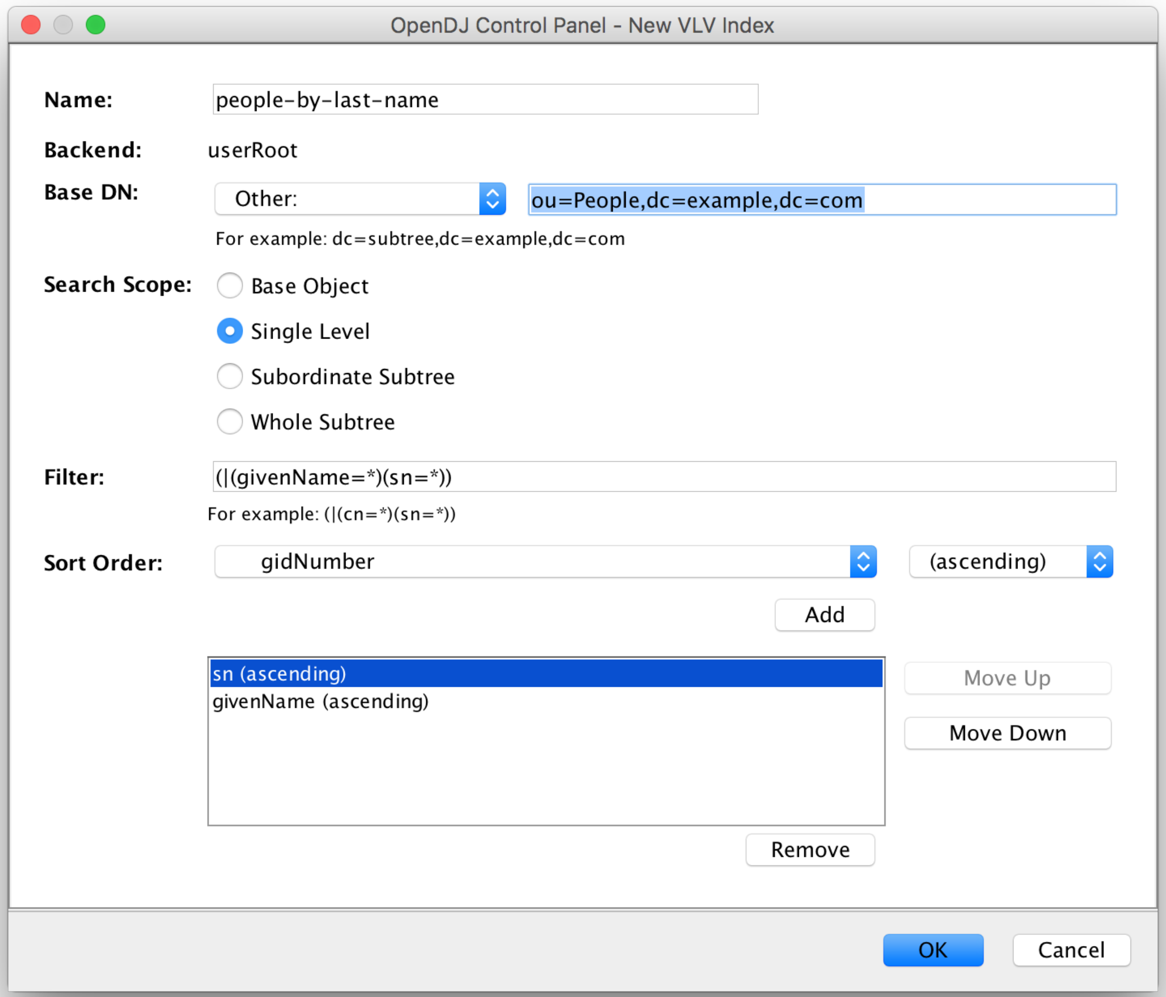
The OpenDJ control panel New Index window does not help you set up extensible matching rule indexes. Use the `dsconfig` command instead.

The following example configures an extensible matching rule index for "later than" and "earlier than" generalized time matching on a `lastLoginTime` attribute in a backend of type `pdb` named `myData`:

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



OpenDJ Control Panel - New VLV Index

Name: people-by-last-name

Backend: userRoot

Base DN: Other: ou=People,dc=example,dc=com

For example: dc=subtree,dc=example,dc=com

Search Scope: Base Object [✓ Single Level] Subordinate Subtree Whole Subtree

Filter: (|(givenName=*)(sn=*))

For example: (|(cn=*)(sn=*))

Sort Order: gidNumber (ascending)

sn (ascending)
givenName (ascending)

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 a backend of type `pdb` named `myData`:

```
$ dsconfig \
 create-backend-vlv-index \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --backend-name myData \
 --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.

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --index cn \
 --start 0 \
 --trustAll
Rebuild Index task 20150219181540575 scheduled to start Feb 19, 2015 6:15:40
```

Rebuild Degraded Indexes

The following example rebuilds degraded indexes immediately with the server online:

```
$ rebuild-index \
 --port 4444 \
 --hostname opendj.example.com \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --baseDN dc=example,dc=com \
  --rebuildDegraded
...
...message="Due to changes in the configuration,
 index dc=com,dc=example_description is currently operating in a degraded state
 and must be rebuilt before it can be used"
...message="Rebuild of all degraded indexes started
 with 177 total entries to process"
..."Rebuild complete. Processed 177 entries in 0 seconds
 (average rate 3160.7/sec)"
...
Rebuild Index task 20151031164835613 has been successfully completed
```

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 directory server must be stopped before you use the `backendstat` command:

```
$ stop-ds
```

The third column of the output is the `Index Valid` column, which is `false` before the rebuild, `true` after:

```
$ 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 --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 OpenDJ directory server:

```
$ start-ds
```

If the newly indexed attribute has already been used, rebuild the index instead of clearing the degraded state.

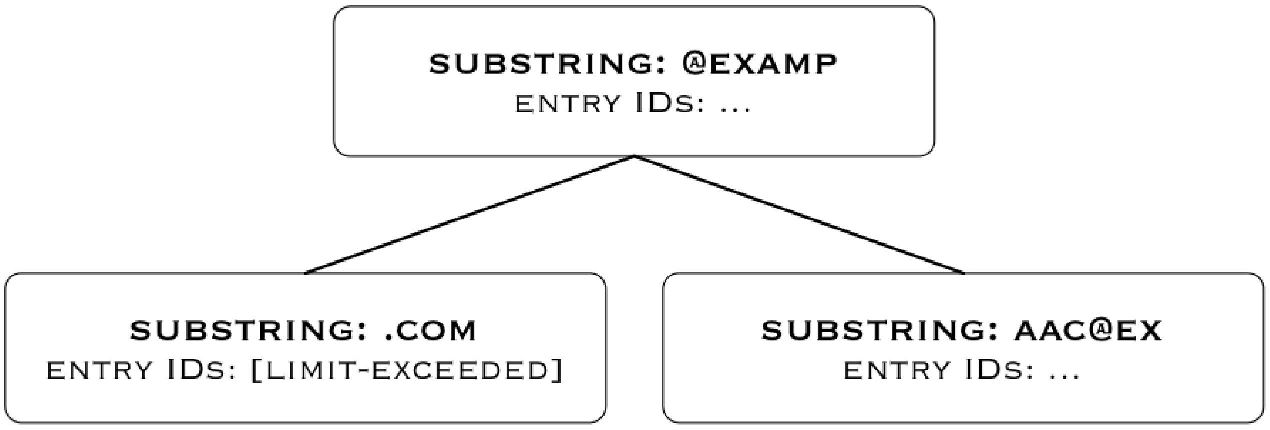
Understanding Index Entry Limits

As described in "About Indexes", an OpenDJ directory server 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.

OpenDJ directory server therefore defines an *index entry limit*. When the number of entry IDs for a key exceeds the limit, OpenDJ directory 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.



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.

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 10,000 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 directory server before you use the `backendstat` command:

```
$ stop-ds
```

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                                ... Index Valid  Record Count  Over Entry
Limit  95%  90%  85%
--------------------------------------...
------------------------------------------------------------
uniqueMember.uniqueMemberMatch            ... true         0             0
0    0    0
mail.caseIgnoreIA5Match                   ... true         10000         0
0    0    0
mail.caseIgnoreIA5SubstringsMatch:6       ... true         31235         15
0    0    0
telephoneNumber....                       ... true         73235         0
0    0    0
telephoneNumber.telephoneNumberMatch      ... true         10000         0
0    0    0
aci.presence                              ... true         0             0
0    0    0
ds-sync-hist....                          ... true         0             0
0    0    0
cn.caseIgnoreMatch                        ... true         10000         0
0    0    0
cn.caseIgnoreSubstringsMatch:6            ... true         86040         0
0    0    0
objectClass.objectIdentifierMatch         ... true         6             4
0    0    0
entryUUID.uuidMatch                       ... true         10002         0
0    0    0
uid.caseIgnoreMatch                       ... true         10000         0
0    0    0
givenName.caseIgnoreMatch                 ... true         8605          0
```

```
0    0    0
givenName.caseIgnoreSubstringsMatch:6 ... true        19629         0
0    0    0
member.distinguishedNameMatch         ... true        0             0
0    0    0
sn.caseIgnoreMatch                    ... true        10000         0
0    0    0
sn.caseIgnoreSubstringsMatch:6        ... true        32217         0
0    0    0
ds-sync-conflict....                  ... true        0             0
0    0    0

Total: 18

Index: /dc=com,dc=example/objectClass.objectIdentifierMatch
Over index-entry-limit keys: [2.5.6.0] [2.5.6.6] [2.5.6.7] [inetorgperson]

Index: /dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6
Over index-entry-limit keys: [.net] [@maild] [aildom] [ain.ne] [domain] [et]
[ildoma]
 [in.net] [ldomai] [maildo] [main.n] [n.net] [net] [omain.] [t]
```

Every user entry has the object classes listed, and every user entry has an email address ending in `@maildomain.net`, so those values are not specific enough to be used in search filters.

3. Start OpenDJ directory server:

```
$ start-ds
```

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, 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name objectClass \
 --set index-entry-limit:10000 \
 --trustAll \
 --no-prompt

$ rebuild-index \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --index objectClass \
 --start 0
Rebuild Index task 20160729123736723 scheduled to start ...
```

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --set index-entry-limit:10000 \
 --trustAll \
 --no-prompt
```

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` (common name) index for completeness and for errors:

```
$ verify-index \
 --baseDN dc=example,dc=com \
 --index cn \
 --clean \
 --countErrors
...msg=Checked 1316 records and found 0 error(s) in 0 seconds
 (average rate 2506.7/sec)
...msg=Number of records referencing more than one entry: 315
...msg=Number of records that exceed the entry limit: 0
...msg=Average number of entries referenced is 1.58/record
...msg=Maximum number of entries referenced by any record is 32
```

Ignore the messages regarding lock tables and cleaner threads. The important information is whether any errors are found in the indexes.

Default Indexes

When you first install OpenDJ directory server and import your data from LDIF, the following indexes are configured.

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
`telephoneNumber`	-	Yes	-	-	Yes	4000
`uid`	-	Yes	-	-	-	4000
`uniqueMember`	-	Yes	-	-	-	4000

When you create a new backend using the `dsconfig` command, OpenDJ directory server creates 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".

Managing Data Replication

OpenDJ 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:

- Set up replication as part of initial installation using OpenDJ control panel, or at any time using command-line tools
- Understand how replication operates in order to configure it appropriately
- Enable, 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

Replication Quick Setup

You can set up replication during installation by using the setup wizard, starting with the Topology Options screen:

- In the Topology Options screen for the first server you set up, select This server will be part of a replication topology.

  If you also choose Configure as Secure, then replication traffic is protected by Transport Layer Security.

- In the Topology Options screen for subsequent servers, select There is already a server in the topology.

  Provide the Host Name, Administration Connector Port number, global Admin User identifier, and Admin Password for the first server.

- When presented with the Create Global Administrator screen, provide a Global Administrator ID and Global Administrator Password.

  The Global Administrator account exists on all servers in the replication topology. The account is stored under `cn=admin data`. It provides an account to administer replication with the same credentials on every server in the topology.

- In the Data Replication screen, select the user and application data base DN(s) to replicate.

  OpenDJ directory server automatically replicates configuration data and directory schema.

Once replication is set up, it works for all the replicas. You can monitor replication status through OpenDJ control panel.

About Replication

Before you take replication further than setting up replication in the setup wizard, read this section to learn more about how OpenDJ replication works.

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. [1] Letting convergence happen over time means that different replicas can be momentarily out of sync, but 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.

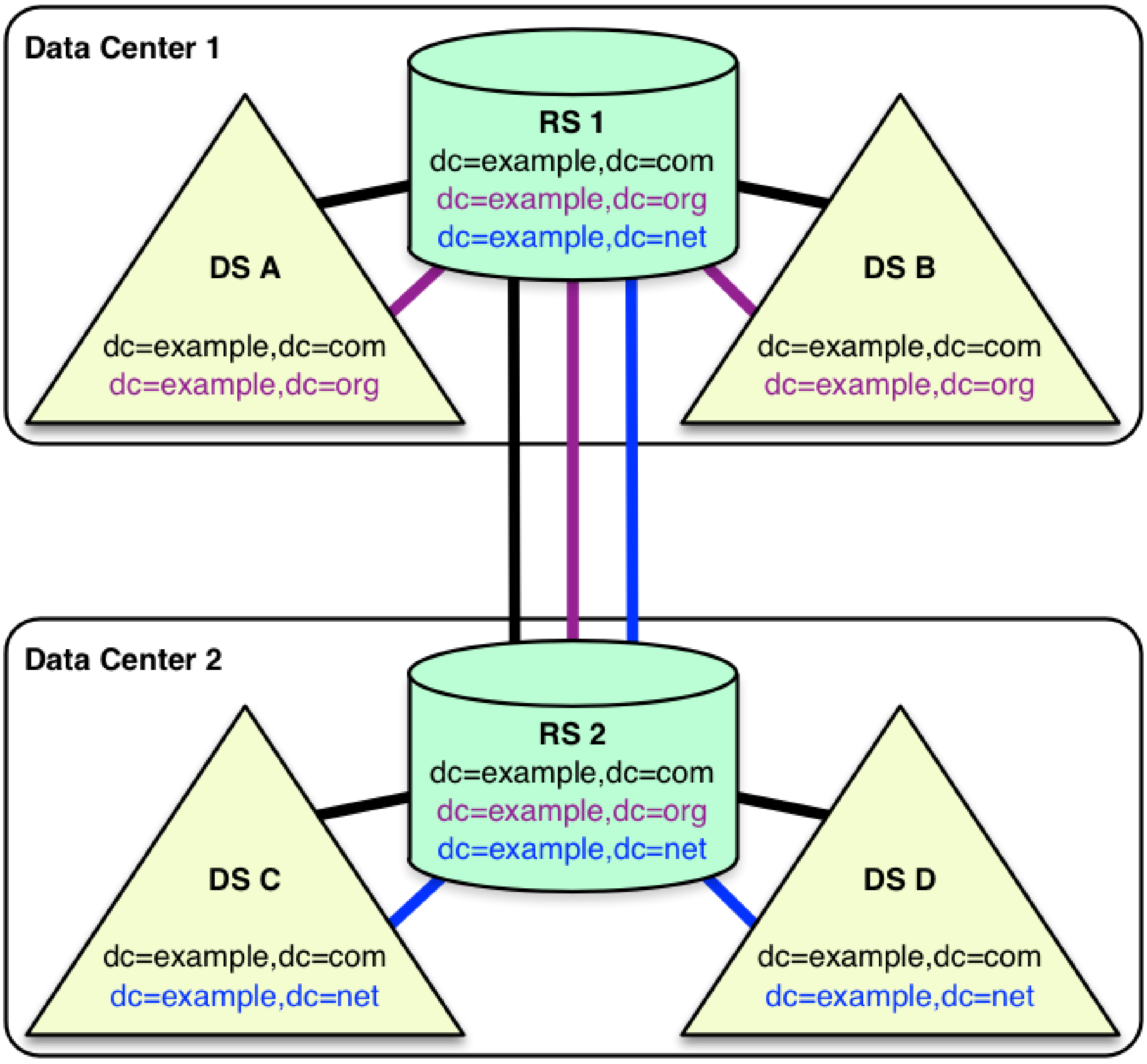
Replication is specific to the OpenDJ directory service. Replication uses a specific protocol that replays update operations quickly, storing enough historical information about the updates to resolve most conflicts automatically. For example, if two client applications separately update a user entry to change the phone number, replication can identify the latest change, and apply it across servers. The historical information needed to resolve these issues is periodically purged to avoid becoming too large. As a directory administrator, you must ensure that you do not purge the historical information more often than you back up your directory data.

Keep server clocks synchronized for your topology. You can use NTP for example. Keeping server clocks synchronized helps prevent issues with SSL connections and with replication itself. Keeping server clocks synchronized also makes it easier to compare timestamps from multiple servers.

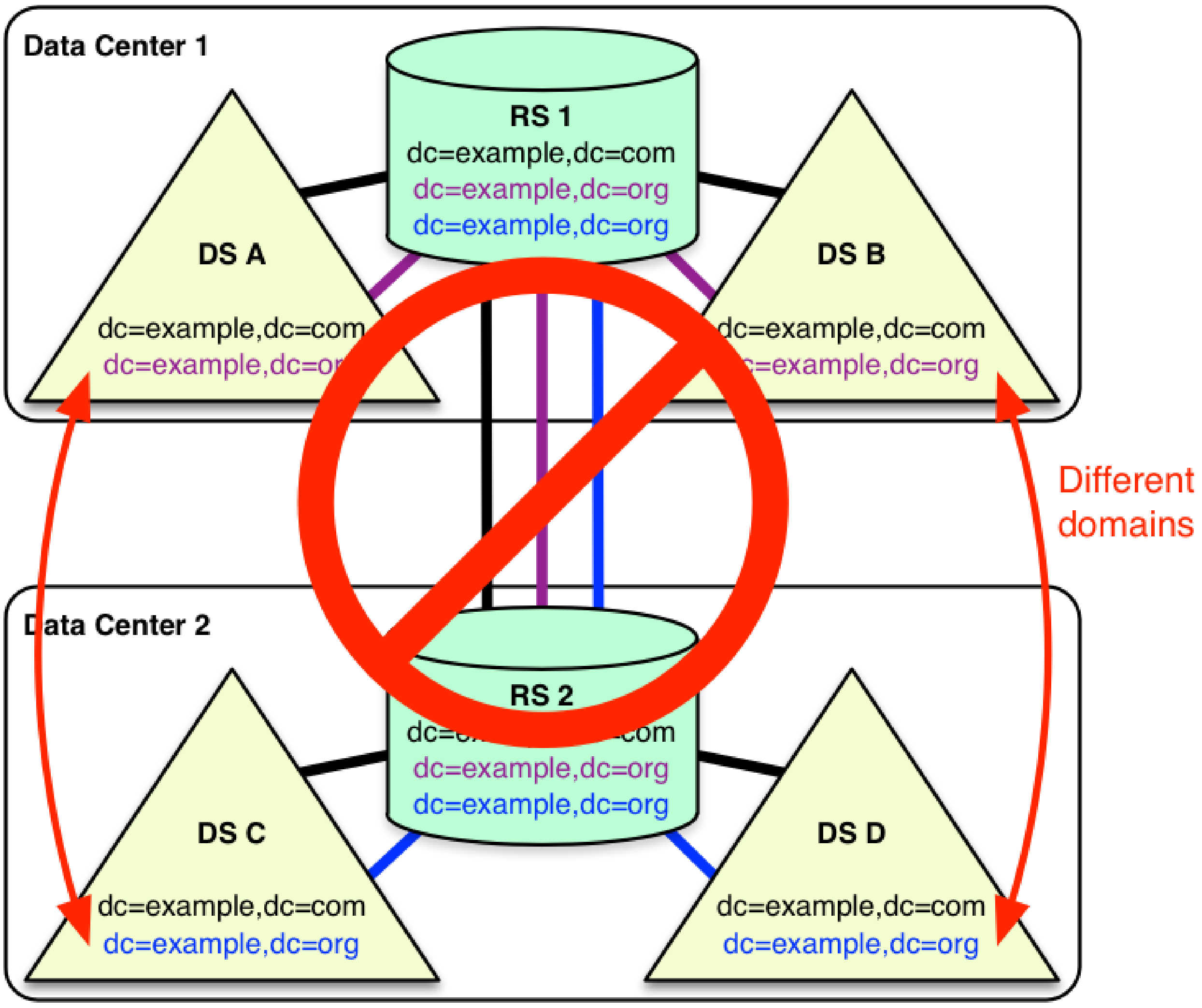
Replication Per Suffix

The primary unit of replication is the suffix, specified by a base DN such as `dc=example,dc=com`.[2] 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. Thus 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.



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 the following diagram.



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 can offer both directory service and also replication service, and the two services are not the same, even if they can run alongside each other in the same OpenDJ server in the same Java Virtual Machine.

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. A replication server manages replication data only, handling replication traffic with 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.

A directory server handles directory data. It responds to requests, stores directory data and historical information. For each replicated suffix, such as `dc=example,dc=com`, `cn=schema` and `cn=admin data`, the directory server publishes changes to a replication server, and subscribes to changes from that replication server. (Directory servers do not publish changes to other directory servers.) 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 to which it can connect, 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. Had the same initial data for the suffix as the directory server
3. If initial data was the same, has all the latest changes from the directory server
4. Runs in the same Java Virtual Machine as the directory server
5. Has 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, OpenDJ 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, and once 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.

Configuring Replication

This section shows how to configure replication with command-line tools, such as the `dsreplication` command, described in dsreplication(1) in the *Reference*.

Enabling Replication

You can start the replication process by using the `dsreplication enable` command:

```
$ dsreplication \
 enable \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
 --host2 opendj2.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --trustAll \
 --no-prompt

Establishing connections ..... Done.
Checking registration information ..... Done.
Updating remote references on server opendj.example.com:4444 ..... Done.
Configuring Replication port on server opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj2.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj2.example.com:4444 ..... Done.
Initializing registration information on server opendj2.example.com:4444 with
 the contents of server opendj.example.com:4444 ..... Done.
Initializing schema on server opendj2.example.com:4444 with the contents of
 server opendj.example.com:4444 ..... Done.

Replication has been successfully enabled.  Note that for replication to
 work you must initialize the contents of the base DN's that are being
  replicated (use dsreplication initialize to do so).

See
```

```
/var/.../opends-replication-7958637258600693490.log
for a detailed log of this operation.
```

To enable secure connections for replication use the `--secureReplication1` and `--secureReplication2` options, which are equivalent to selecting Configure as Secure in the replication topology options screen of the setup wizard.

As you see in the command output, replication is set up to function once enabled. You must, however, initialize replication in order to start the process.

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.

If you need to add another OpenDJ directory server to participate in replication, use the `dsreplication enable` with the new server as the second server.

Initializing Replicas

You can initialize replication between servers by performing initialization over the network after you have enabled replication, or by importing the same LDIF data on all servers and then enabling replication. You can also add a new server by restoring a backup from an existing replica onto the new server and then enabling 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. Enable replication on all servers.

   See "Enabling 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

Initializing base DN dc=example,dc=com with the contents from
 opendj.example.com:4444: 160 entries processed (100 % complete).
Base DN initialized successfully.

See
/var/.../opends-replication-5020375834904394170.log
for a detailed log of this operation.
```

To Initialize All Servers From the Same LDIF

This procedure can be useful when you are starting with a large amount of directory data that is available locally to all directory servers.

1. Enable replication for all servers.

   IMPORTANT Enabling replication means overwriting 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 enabled, however, the administrative data is also shared through replication. If you use data confidentiality to protect data stored on disk, then replication must be enabled before you import data to allow the replicas to share rather than overwrite each others' encryption keys.

   See "Enabling 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 a server in the existing topology.

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. Enable replication on the new replica:

```
$ dsreplication \
 enable \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
 --host2 opendj3.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --trustAll \
 --no-prompt

Establishing connections ..... Done.
Checking registration information ..... Done.
Updating remote references on server opendj.example.com:4444 ..... Done.
Configuring Replication port on server opendj3.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj3.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj2.example.com:4444 ..... Done.
Updating remote references on server opendj2.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj3.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj3.example.com:4444 ..... Done.
```

```
Updating replication configuration for baseDN cn=schema on server
 opendj2.example.com:4444 ..... Done.
Initializing registration information on server opendj3.example.com:4444 with
 the contents of server opendj.example.com:4444 ..... Done.

Replication has been successfully enabled.  Note that for replication to
 work you must initialize the contents of the base DN's that are being
 replicated (use dsreplication initialize to do so).

See
/var/.../opends-replication-1672058070147419978.log
for a detailed log of this operation.
```

Contrary to the message from the command, you do not need to use the `dsreplication initialize` command at this point.

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 \
    --baseDN dc=example,dc=com \
    --hostname opendj.example.com \
    --port 4444 \
    --trustAll \
    --no-prompt

   Preparing base DN dc=example,dc=com to be initialized externally ..... Done.

   Now you can proceed to the initialization of the contents of the base DNs on
   all the replicated servers.  You can use the command import-ldif or the binary
   copy to do so.  You must use the same LDIF file or binary copy on each server.

   When the initialization is completed you must use the subcommand
   'post-external-initialization' for replication to work with the new base DNs
   contents.
   ```

   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 Stand-alone 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 \
 --baseDN dc=example,dc=com \
 --hostname opendj.example.com \
 --port 4444 \
 --trustAll \
 --no-prompt

Updating replication information on base DN dc=example,dc=com ..... Done.


Post initialization procedure completed successfully.
```

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`.

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 `dsconfig` command.

WARNING Do not allow modifications on the replica for which replication is disabled, as no record of such changes is kept, and the changes cause replication to diverge.

1. Disable the multimaster synchronization provider:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

2. (Optional) When you are ready to resume replication, enable the multimaster synchronization provider:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --port 4444 \
 --hostname opendj2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

To Stop Replication Permanently For a Replica

If you need to stop a server from replicating permanently, for example in preparation to remove a server, you can do so with the `dsreplication disable` command.

1. Stop replication using the `dsreplication disable` command:

```
$ dsreplication \
 disable \
 --disableAll \
 --port 4444 \
 --hostname opendj2.example.com \
 --adminUID admin \
 --adminPassword password \
 --trustAll \
 --no-prompt
Establishing connections ..... Done.
Disabling replication on base DN cn=admin data of server
 opendj2.example.com:4444 ..... Done.
Disabling replication on base DN dc=example,dc=com of server
 opendj2.example.com:4444 ..... Done.
Disabling replication on base DN cn=schema of server
 opendj2.example.com:4444 ..... Done.
Disabling replication port 8989 of server
 opendj2.example.com:4444 ..... Done.
Removing registration information ..... Done.
Removing truststore information ..... Done.

See
/var/.../opends-replication-125248191132797765.log
for a detailed log of this operation.
```

   The `dsreplication disable` as shown completely removes the replication configuration information from the server.

2. (Optional) If you want to restart replication for the server, you need to run the

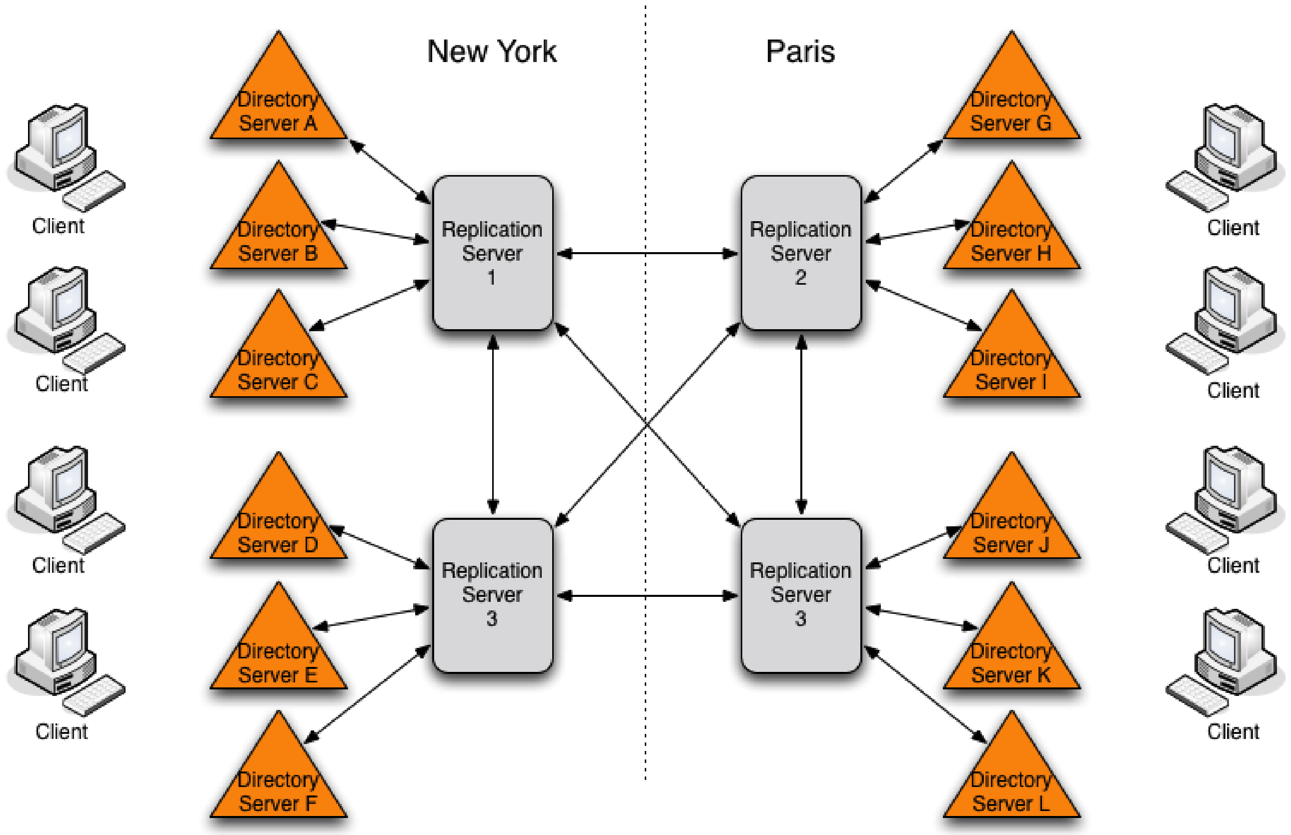
`dsreplication enable` and `dsreplication initialize` commands again.

Standalone Replication Servers

Replication in OpenDJ is designed to be both easy to implement in environments with a few servers, and also scalable in environments with many servers. You can enable the replication service on each OpenDJ directory server in your deployment, for example, to limit the number of servers you deploy. Yet in a large deployment, you can use standalone replication servers—OpenDJ servers that do nothing but relay replication messages—to configure (and troubleshoot) the replication service separately from the directory service. You only need a few standalone replication servers publishing changes to serve many directory servers subscribed to the changes. Furthermore, replication is designed such that you need only connect a directory server to the nearest replication server for the directory server to replicate with all others in your topology. Yet 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, $\text{Total}_{\text{conn}}$ is expressed as follows.

$\text{Total}_{\text{conn}} = (N_{RS} * (N_{RS} -1))/2 + N_{DS}$ 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 $\text{Total}_{\text{conn}}$ is three with no standalone servers. However, if you have two data centers, and need 12 directory servers, then with no standalone directory servers $\text{Total}_{\text{conn}}$ is (12 * 11)/2 or 66. Yet, with four standalone replication servers, and 12 standalone directory servers, $\text{Total}_{\text{conn}}$ is (4 * 3)/2 + 12, or 18, with only four of those connections needing to go 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.)



TIP

If you set up OpenDJ directory server to replicate by using the Quick Setup wizard, then the wizard activated the replication service for that server. You can turn off the replication service on OpenDJ directory server, and then configure the server to work with a separate, standalone replication server instead. Start by using the `dsreplication disable --disableReplicationServer` command to turn off the replication service on the server.

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.example.com`. The directory servers are `opendj.example.com` and `opendj2.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. Enable replication with `--noReplicationServer` or `--onlyReplicationServer` options:

```
$ dsreplication \
 enable \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --noReplicationServer1 \
 --host2 rs.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --onlyReplicationServer2 \
 --trustAll \
 --no-prompt
Establishing connections ..... Done.
Only one replication server will be defined for the following base DN's:
dc=example,dc=com
It is recommended to have at least two replication servers (two changelogs) to
avoid a single point of failure in the replication topology.

Checking registration information ..... Done.
Configuring Replication port on server rs.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 rs.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj.example.com:4444 ..... Done.
Initializing registration information on server rs.example.com:4444 with
 the contents of server opendj.example.com:4444 ..... Done.

Replication has been successfully enabled.  Note that for replication to work
 you must initialize the contents of the base DN's that are being
 replicated (use dsreplication initialize to do so).

See
/var/.../opends-replication-1720959352638609971.log
for a detailed log of this operation.

$ dsreplication \
 enable \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj2.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --noReplicationServer1 \
 --host2 rs.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --onlyReplicationServer2 \
 --trustAll \
 --no-prompt

Establishing connections ..... Done.
Only one replication server will be defined for the following base DN's:
dc=example,dc=com
It is recommended to have at least two replication servers (two changelogs) to
avoid a single point of failure in the replication topology.

Checking registration information ..... Done.
Updating remote references on server rs.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj2.example.com:4444 ..... Done.
Updating registration configuration on server
 rs.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj.example.com:4444 ..... Done.
Initializing registration information on server opendj2.example.com:4444 with
 the contents of server rs.example.com:4444 ..... Done.

Replication has been successfully enabled.  Note that for replication to work
 you must initialize the contents of the base DN's that are being
 replicated (use dsreplication initialize to do so).

See
/var/folders/.../opends-replication-5893037538856033562.log
for a detailed log of this operation.
```

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

Initializing base DN dc=example,dc=com with the contents from
 opendj.example.com:4444: 160 entries processed (100 % complete).
Base DN initialized successfully.

See
/var/.../opends-replication-7677303986403997574.log
for a detailed log of this operation.
```

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. Enable replication with the appropriate `--noReplicationServer` option:

```
$ dsreplication \
 enable \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 master.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

Establishing connections ..... Done.
Checking registration information ..... Done.
Updating remote references on server master.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com
 on server master.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com
 on server ds-only.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com
 on server master2.example.com:4444 ..... Done.
Updating remote references on server master2.example.com:4444 ..... Done.
Updating registration configuration
 on server master.example.com:4444 ..... Done.
Updating registration configuration
 on server ds-only.example.com:4444 ..... Done.
Updating registration configuration
 on server master2.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema
 on server master.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema
 on server ds-only.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema
 on server master2.example.com:4444 ..... Done.
Initializing registration information on server ds-only.example.com:4444
 with the contents of server master.example.com:4444 ..... Done.
Initializing schema on server ds-only.example.com:4444
 with the contents of server master.example.com:4444 ..... Done.

Replication has been successfully enabled.  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).

See
/var/.../opendj-replication-859181866587327450.log
for a detailed log of this operation.
```

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 master.example.com \
 --portSource 4444 \
 --hostDestination ds-only.example.com \
 --portDestination 4444 \
 --trustAll \
 --no-prompt

Initializing base DN dc=example,dc=com with the contents
 from master.example.com:4444:
0 entries processed (0 % complete).
176 entries processed (100 % complete).
Base DN initialized successfully.

See
/var/.../opendj-replication-4326340645155418876.log
for a detailed log of this operation.
```

5. If you want to make the directory server replica read-only for client application traffic, see "Read-Only Replicas".

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 `opendj2.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, such as all the replicas and replication servers in each data center being in 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj2.example.com \
 --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 \
 --port 4444 \
 --hostname rs.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set group-id:1 \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-replication-server-prop \
 --port 4444 \
 --hostname rs2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set group-id:2 \
 --trustAll \
 --no-prompt
```

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 \
 --port 4444 \
 --hostname opendj2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set writability-mode:internal-only \
 --trustAll \
 --no-prompt
```

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 in the topology 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 having received the update, or to wait for all directory servers to have replayed the update.

As you might imagine, assured replication is theoretically safer than standard replication, yet it is also slower, potentially waiting for a timeout before failing when the network or other servers are down.

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 also set the safe data level:

  ```
  $ dsconfig \
   set-replication-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 assured-type:safe-data \
 --set assured-sd-level:1 \
 --trustAll \
 --no-prompt

$ dsconfig \
 set-replication-domain-prop \
 --port 4444 \
 --hostname opendj2.example.com \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set assured-type:safe-read \
 --trustAll \
 --no-prompt
```

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.

Subtree Replication

OpenDJ can perform subtree replication, for example, replicating 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.

For example, in this case 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. Then you replicate ou=People,dc=example,dc=com in its own topology.

Fractional Replication

OpenDJ can perform fractional replication, whereby you specify the attributes to include in or to exclude from the replication process.

You set fractional replication configuration as fractional-include or fractional-exclude properties for 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, OpenDJ 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --trustAll \
 --no-prompt \
 --set \
 fractional-include:inetorgperson:cn,givenname,mail,mobile,sn,telephonenumber
```

As another example, you might exclude a custom attribute called `sessionToken` from being replicated:

```
$ dsconfig \
 set-replication-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 fractional-exclude:*:sessionToken \
 --trustAll \
 --no-prompt
```

This last example only works if you first define a `sessionToken` attribute in the directory server schema.

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.[3]

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).

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

OpenDJ directory servers without replication cannot expose an external change log. The OpenDJ server that exposes the change log must function both as a directory server, and also as a replication server for the suffix whose changes you want logged.

- Enable replication without using the `--noReplicationServer` or `--onlyReplicationServer` options.

  With replication enabled, 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 \
   "(objectclass=*)" \
   \* +
  dn: cn=changelog
  cn: changelog
  objectClass: top
  objectClass: container
  subschemaSubentry: cn=schema
  hasSubordinates: false
  ```

```
entryDN: 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

NOTE This feature is new in OpenDJ directory server 3.5.

OpenDJ directory server does 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:

```
$ strings /path/to/opendj/changelogDb/*/*/head.log | grep bjensen | sort | uniq
bjensen@example.com0B
bjensen@example.org0B
uid=bjensen,ou=People,dc=example,dc=com
```

In addition to preventing read access by other users as described in "Set Up a System Account for OpenDJ Directory Server", you can configure confidentiality for external change log data. When confidentiality is enabled, OpenDJ directory 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 "Protect Directory Server Network Connections" to protect data sent over the network.

OpenDJ directory server encrypts 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", 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 directory server does 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.

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.

   Notice the value of the `changeLogCookie` attribute for the last of the two changes:

   ```
   $ ldapsearch \
    --baseDN cn=changelog \
    --port 1389 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --control "1.3.6.1.4.1.26027.1.5.4:false" \
    "(objectclass=*)" \
    \* +
   dn: cn=changelog
   cn: changelog
   objectClass: top
   objectClass: container
   subschemaSubentry: cn=schema
   hasSubordinates: true
   entryDN: cn=changelog

   # Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4):
    dc=example,dc=com:0000013087cbc28212d100000001;
   dn: replicationCSN=0000013087cbc28212d100000001,dc=example,dc=com,cn=changelog
   targetDN: cn=arsene lupin,ou=special users,dc=example,dc=com
   changeNumber: 0
   changes::
   b2JqZWN0Q2xhc3M6IHBlcnNvbgpvYmplY3RDbGFzczogdG9wCmNuOiBBcnNlbmUgTHVwaW
    4KdGVsZXBob25lTnVtYmVyOiArMzMgMSAyMyA0NSA2NyA4OQpzbjogTHVwaW4KZW50cnlVVUlEOiA5
   M
    GM3MTRmNy00ODZiLTRkNDctOTQwOS1iNDRkMTlkZWEzMWUKY3JlYXRlVGltZXN0YW1wOiAyMDExMDY
   x
    MzA2NTg1NVoKY3JlYXRvcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290IEROcyxjbj
   1
    jb25maWcK
   changeType: add
   changeTime: 20110613065855Z
   objectClass: top
   objectClass: changeLogEntry
   targetEntryUUID: 90c714f7-486b-4d47-9409-b44d19dea31e
   replicationCSN: 0000013087cbc28212d100000001
   numSubordinates: 0
   replicaIdentifier: 4817
   changeLogCookie: dc=example,dc=com:0000013087cbc28212d100000001;
   changeInitiatorsName: cn=Directory Manager,cn=Root DNs,cn=config
   subschemaSubentry: cn=schema
   hasSubordinates: false
   entryDN:
   replicationCSN=0000013087cbc28212d100000001,dc=example,dc=com,cn=change
    log
   ```

```
# Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4):
 dc=example,dc=com:0000013087cbc34a12d100000002;
dn: replicationCSN=0000013087cbc34a12d100000002,dc=example,dc=com,cn=changelog
targetDN: cn=horace velmont,ou=special users,dc=example,dc=com
changeNumber: 0
changes::
b2JqZWN0Q2xhc3M6IHBlcnNvbgpvYmplY3RDbGFzczogdG9wCmNuOiBIb3JhY2UgVmVsbW
 9udAp0ZWxlcGhvbmVOdW1iZXI6ICszMyAxIDEyIDIzIDM0IDQ1CnNuOiBWZWxtb250CmVudHJ5VVVJ
R
 DogNmIyMjQ0MGEtNzZkMC00MDMxLTk0YjctMzViMWQ4NmYwNjdlCmNyZWF0ZVRpbWVzdGFtcDogMjA
x
 MTA2MTMwNjU4NTVaCmNyZWF0b3JzTmFtZTogY249RGlyZWN0b3J5IE1hbmFnZXIsY249Um9vdCBETn
M
 sY249Y29uZmlnCg==
changeType: add
changeTime: 20110613065855Z
objectClass: top
objectClass: changeLogEntry
targetEntryUUID: 6b22440a-76d0-4031-94b7-35b1d86f067e
replicationCSN: 0000013087cbc34a12d100000002
numSubordinates: 0
replicaIdentifier: 4817
changeLogCookie: dc=example,dc=com:0000013087cbc34a12d100000002;
changeInitiatorsName: cn=Directory Manager,cn=Root DNs,cn=config
subschemaSubentry: cn=schema
hasSubordinates: false
entryDN:
replicationCSN=0000013087cbc34a12d100000002,dc=example,dc=com,cn=change
 log
```

In this example, two new users were added to another replica before the change log request was made.

Here the changes are base64-encoded, so you can decode them using the base64 command:

```
$ base64 decode --encodedData b2JqZW...ZmlnCg==
objectClass: person
objectClass: top
cn: Horace Velmont
telephoneNumber: +33 1 12 23 34 45
sn: Velmont
entryUUID: 6b22440a-76d0-4031-94b7-35b1d86f067e
createTimestamp: 20110613065855Z
creatorsName: cn=Directory Manager,cn=Root DNs,cn=config
```

2. For the next search, provide the cookie to start reading where you left off last time.

   In this example, a description was added to Babs Jensen’s entry:

```
$ ldapsearch \
 --baseDN cn=changelog \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --control "1.3.6.1.4.1.26027.1.5.4:false:dc=example, \
  dc=com:0000013087cbc34a12d100000002;" \
 "(objectclass=*)" \
 \* +
dn: cn=changelog
cn: changelog
objectClass: top
objectClass: container
subschemaSubentry: cn=schema
hasSubordinates: true
entryDN: cn=changelog

# Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4):
 dc=example,dc=com:0000013087d7e27f12d100000003;
dn: replicationCSN=0000013087d7e27f12d100000003,dc=example,dc=com,cn=changelog
targetDN: uid=bjensen,ou=people,dc=example,dc=com
changeNumber: 0
changes::
YWRkOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogQSB0aGlyZCBjaGFuZ2UKLQpyZXBsYW
 NlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJv
b
 3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IG1vZGlmeVRpbWVzdGFtcAptb2RpZnlUaW1lc3RhbXA
6
 IDIwMTEwNjEzMDcxMjEwWgotCg==
changeType: modify
changeTime: 20110613071210Z
objectClass: top
objectClass: changeLogEntry
targetEntryUUID: fc252fd9-b982-3ed6-b42a-c76d2546312c
replicationCSN: 0000013087d7e27f12d100000003
numSubordinates: 0
replicaIdentifier: 4817
changeLogCookie: dc=example,dc=com:0000013087d7e27f12d100000003;
changeInitiatorsName: cn=Directory Manager,cn=Root DNs,cn=config
subschemaSubentry: cn=schema
hasSubordinates: false
entryDN:
replicationCSN=0000013087d7e27f12d100000003,dc=example,dc=com,cn=change
 log
```

If we base64-decode the changes, we see the following:

```
$ base64 decode --encodedData YWRkO...gotCg==
add: description
```

```
description: A third change
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20110613071210Z
-
```

3. If for some reason you lose the cookie, you can 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. (Optional) 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:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: changelog-read

Processing MODIFY request for cn=My App,ou=Apps,dc=example,dc=com
MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

4. Test that the user can read the changelog:

```
$ ldapsearch \
 --baseDN cn=changelog \
 --port 1389 \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --control "1.3.6.1.4.1.26027.1.5.4:false" \
 "(objectclass=*)" \
 \* +
dn: cn=changelog
objectClass: top
objectClass: container
cn: changelog
subschemaSubentry: cn=schema
hasSubordinates: true
entryDN: cn=changelog

# Public changelog exchange control(1.3.6.1.4.1.26027.1.5.4): dc=example,dc
=com:...;
dn: replicationCSN=0000015530c8479f20d800000001,dc=example,dc=com,cn=changelog
objectClass: top
objectClass: changeLogEntry
...
```

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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

You can 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 OpenDJ from logging changes by disabling the domain:

```
$ 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
```

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.

   See /path/to/opendj-replication-....log
   for a detailed log of this operation.
   ```

   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.

Recovering From User Error

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 also provides 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:

  ```
  # 30/Apr/2014:16:23:29 +0200; conn=7; op=10
  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: 20140430142329Z

  # 30/Apr/2014:16:23:46 +0200; conn=7; op=14
  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: 20140430142346Z

# 30/Apr/2014:16:24:53 +0200; conn=7; op=27
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 disabling 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 keeps a copy of the last configuration that OpenDJ could use to start the server in the file `/path/to/opendj/config/config.ldif.startok`.

  OpenDJ also backs 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.

[1] Assured replication can require, however, that the convergence happen before the client application is notified that the operation was successful.

[2] 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.

[3] 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.

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 directory server products, see "Managing Directory Data" instead. In this chapter you will learn to:

- Create backup archives
- Restore data from backup archives

OpenDJ 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, as well as backing up server configuration information.

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.*

Backing Up Directory Data

A `bak/` directory is provided when you install OpenDJ, as a location to save binary backups. When you create a backup, the `bak/backup.info` contains information about the archive. This is acceptable if you have only one backend to back up. Each `backup.info` file only contains information about one backend, however. If you have more than one backend, then use a separate backup directory for each backend in order to have separate `backup.info` files for each backend ID.

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 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 \
     --port 4444 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password \
     --backendID userRoot \
     --backupDirectory /path/to/opendj/bak/userRoot \
     --start 0
    Backup task 20110613143715983 scheduled to start Jun 13, 2011 2:37:15 PM
    CEST
    ```

  - Stop the server to back up Example.com data offline.

    The following example stops OpenDJ, runs offline backup, and starts the server after backup has completed:

    ```
    $ stop-ds
    Stopping Server...

    [13/Jun/2011:14:31:00 +0200] category=BACKEND severity=NOTICE msgID=9896306
     msg=The backend userRoot is now taken offline
    [13/Jun/2011:14:31:00 +0200] category=CORE severity=NOTICE msgID=458955
     msg=The Directory Server is now stopped
    $ backup --backendID userRoot -d /path/to/opendj/bak
    ```

```
[13/Jun/2011:14:33:48 +0200] category=TOOLS severity=NOTICE msgID=10944792
 msg=Starting backup for backend userRoot
...
[13/Jun/2011:14:33:48 +0200] category=TOOLS severity=NOTICE msgID=10944795
 msg=The backup process completed successfully
$ start-ds
... The Directory Server has started successfully
```

- Back up all user data on the server.

  The following example requests an online backup task that starts immediately, backing up all backends:

  ```
  $ backup \
   --port 4444 \
   --bindDN "cn=Directory Manager" \
   --bindPassword password \
   --backUpAll \
   --backupDirectory /path/to/opendj/bak/userRoot \
   --start 0
  Backup task 20110613143801866 scheduled to start Jun 13, 2011 2:38:01 PM
  CEST
  ```

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 \
   --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
  Recurring Backup task BackupTask-988d6adf-4d65-44bf-8546-6ea74a2480b0
  scheduled successfully
  ```

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 OpenDJ 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 \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backupDirectory /path/to/opendj/bak/userRoot \
 --backendID userRoot \
 --incremental \
 --recurringTask "00 03 * * *" \
 --completionNotify diradmin@example.com \
 --errorNotify diradmin@example.com
Recurring Backup task BackupTask-6988c19d-9afc-4f50-89b7-d3e167255d3e
scheduled successfully
```

Restoring Directory Data From Backup

When you restore data, the procedure to follow depends on whether the OpenDJ directory server is replicated.

To Restore a Stand-alone Server

To restore OpenDJ 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, and also setting a start time for the task by using the --start option.

To restore data when OpenDJ directory server is stopped, 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 OpenDJ, restores data offline from one of the available backups, and then starts the server after the restore is complete:

    ```
    $ stop-ds
    Stopping Server...

    [13/Jun/2011:15:44:06 +0200] category=BACKEND severity=NOTICE msgID=9896306
     msg=The backend userRoot is now taken offline
    [13/Jun/2011:15:44:06 +0200] category=CORE severity=NOTICE msgID=458955
     msg=The Directory Server is now stopped
    $ restore --backupDirectory /path/to/opendj/bak/userRoot --listBackups
    Backup ID:          20110613080032
    Backup Date:        13/Jun/2011:08:00:45 +0200
    ```

```
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none
$ restore --backupDirectory /path/to/opendj/bak/userRoot --backupID
20110613080032
[13/Jun/2011:15:47:41 +0200] ... msg=Restored: 00000000.jdb (size 341835)
$ start-ds
... The Directory Server has started successfully
```

- Schedule the restore as a task to begin immediately.

  The following example requests an online restore task, scheduled to start immediately:

```
$ restore \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backupDirectory /path/to/opendj/bak/userRoot \
 --backupID 20110613080032 \
 --start 0
Restore task 20110613155052932 scheduled to start Jun 13, 2011 3:50:52 PM
CEST
```

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, enable replication between the new replica server and a server from the existing topology as described in "Enabling Replication".

If the backup is not encrypted, you can skip this step.

This step initiates OpenDJ'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, enable 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:

```
$ stop-ds
Stopping Server...

[13/Jun/2011:15:44:06 +0200] category=BACKEND severity=NOTICE msgID=9896306
 msg=The backend userRoot is now taken offline
[13/Jun/2011:15:44:06 +0200] category=CORE severity=NOTICE msgID=458955
 msg=The Directory Server is now stopped
$ restore --backupDirectory /path/to/opendj/bak/userRoot --listBackups
Backup ID:          20110613080032
Backup Date:        13/Jun/2011:08:00:45 +0200
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none
$ restore --backupDirectory /path/to/opendj/bak/userRoot --backupID
20110613080032
[13/Jun/2011:15:47:41 +0200] ... msg=Restored: 00000000.jdb (size 341835)
$ start-ds
... The Directory Server has started successfully
```

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.

About OpenDJ Password Policies

OpenDJ password policies govern not only passwords, but also account lockout, and how OpenDJ provides notification about account status.

OpenDJ supports password policies as part of the server configuration, and also subentry password policies as part of the (replicated) user data.

Server-Based Password Policies

You manage server-based password policies in the OpenDJ 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, OpenDJ includes two password policy configurations, one default for all users, and another for directory root DN users, such as `cn=Directory Manager`. You can see all the default password policy settings using the `dsconfig` command as follows:

```
$ dsconfig \
 get-password-policy-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --advanced
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-1
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                  : 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 "Password Policy" in the *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.

Yet a few basic protections are configured by default. When you import LDIF with `userPassword` values, OpenDJ hashes the values before storing them. When a user provides a password value during a bind for example, the server hashes the value provided to compared it with the stored value. Even the directory manager cannot see 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: {SSHA}QWAtw8ch/9850HNFRRqLNMIQc1YhxCnOoGmk1g==
```

In addition, users can change their passwords provided you have granted them access to do so. OpenDJ 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.

Subentry-Based Password Policies

You manage subentry password policies by adding the subentries alongside the user data. Thus, OpenDJ can replicate subentry password policies across servers. 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 thus inherited from the default password policy.

As a result, the following Internet-Draft password policy attributes override the default password policy when you set them in the subentry:

- `pwdAllowUserChange`, corresponding to the OpenDJ password policy property `allow-user-password-changes`
- `pwdMustChange`, corresponding to the OpenDJ password policy property `force-change-on-reset`
- `pwdGraceAuthNLimit`, corresponding to the OpenDJ password policy property `grace-login-count`
- `pwdLockoutDuration`, corresponding to the OpenDJ password policy property `lockout-duration`
- `pwdMaxFailure`, corresponding to the OpenDJ password policy property `lockout-failure-count`
- `pwdFailureCountInterval`, corresponding to the OpenDJ password policy property `lockout-failure-expiration-interval`
- `pwdMaxAge`, corresponding to the OpenDJ password policy property `max-password-age`
- `pwdMinAge`, corresponding to the OpenDJ password policy property `min-password-age`
- `pwdAttribute`, corresponding to the OpenDJ password policy property `password-attribute`
- `pwdSafeModify`, corresponding to the OpenDJ password policy property `password-change-requires-current-password`
- `pwdExpireWarning`, corresponding to the OpenDJ password policy property `password-expiration-warning-interval`
- `pwdInHistory`, corresponding to the OpenDJ password policy property `password-history-count`

The following Internet-Draft password policy attributes are not taken into account by OpenDJ:

- `pwdCheckQuality`, as OpenDJ has password validators. You can set password validators to use in the default password policy.
- `pwdMinLength`, as this is handled by the length-based password validator. You can configure this as part of the default password policy.

- `pwdLockout`, as OpenDJ 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 directory 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 OpenDJ directory 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".

Which Password Policy Applies

The password policy that applies to a user is identified by the operational attribute, `pwdPolicySubentry`:

```
$ 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
```

The default global access control instructions prevent this operational attribute from being visible to normal users, so examples show it being accessed by the Directory Manager user.

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.

To Adjust the Default Password Policy

You can reconfigure the default password policy, for example, to enforce password expiration, check that passwords do not match dictionary words, and prevent password reuse. This default policy is a server-based password policy.

1. Enable the appropriate password validator:

```
$ dsconfig \
 set-password-validator-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --validator-name Dictionary \
 --set enabled:true \
 --set check-substrings:true \
 --set min-substring-length:4 \
 --trustAll \
 --no-prompt
```

2. Apply the changes to the default password policy:

```
$ dsconfig \
 set-password-policy-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set max-password-age:90d \
 --set min-password-age:4w \
 --set password-history-count:7 \
 --set password-validator:Dictionary \
 --trustAll \
 --no-prompt
```

3. Check your work:

```
$ dsconfig \
 get-password-policy-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy"
Property                                  : Value(s)
------------------------------------------:-------------------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-user-password-changes               : true
default-password-storage-scheme           : Salted SHA-1
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                          : 12 w 6 d
max-password-reset-age                    : 0 s
min-password-age                          : 4 w
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                        : Dictionary
previous-last-login-time-format           : -
require-change-by-time                    : -
require-secure-authentication             : false
require-secure-password-changes           : false
```

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "New Account Password Policy" \
 --set default-password-storage-scheme:"Salted SHA-1" \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "New Account Password Policy"
Property                                  : Value(s)
------------------------------------------:-------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-user-password-changes               : true
default-password-storage-scheme           : Salted SHA-1
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, you might want to change the user's policy again when the new user successfully updates the password.

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 /path/to/subentry-pwp.ldif
dn: cn=Subentry Password Policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: pwdPolicy
cn: Subentry Password Policy
pwdAttribute: userPassword
pwdLockout: TRUE
pwdMaxFailure: 3
pwdFailureCountInterval: 300
pwdLockoutDuration: 300
pwdAllowUserChange: TRUE
pwdSafeModify: TRUE
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }
```

2. Add the policy to the directory:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd \
 --filename /path/to/subentry-pwp.ldif
Processing ADD request for cn=Subentry Password Policy,dc=example,dc=com
ADD operation successful for DN cn=Subentry Password Policy,dc=example,dc=com
```

3. Check that the policy applies as specified.

   In the 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,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
```

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 subtree 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 further 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.

To Assign a Password Policy to a User

1. Prevent users from selecting their own password policy:

```
$ cat protectpwp.ldif
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///uid=*,ou=People,dc=example,dc=com")(targetattr =
 "ds-pwp-password-policy-dn")(version 3.0;acl "Cannot choose own pass
 word policy";deny (write)(userdn = "ldap:///self");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename protectpwp.ldif
Processing MODIFY request for ou=People,dc=example,dc=com
MODIFY operation successful for DN ou=People,dc=example,dc=com
```

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 "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd \
 --filename newuser.ldif
Processing ADD request for uid=newuser,ou=People,dc=example,dc=com
ADD operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

3. Check your work:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --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

1. 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=Pass
 word Policies,cn=config
subtreeSpecification: { base "ou=People", specificationFilter "(isMemberOf=
 cn=Directory Administrators,ou=Groups,dc=example,dc=com)"}

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd \
 --filename pwp-coll.ldif
Processing ADD request for cn=Password Policy for Dir Admins,dc=example,dc=com
ADD operation successful for DN cn=Password Policy for Dir
 Admins,dc=example,dc=com
```

2. Check your work:

```
$ 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=Root Password Policy,cn=Password Policies,cn=config
```

To Assign Password Policy for an Entire Branch

You can use a collective attribute to assign a password policy to the entries under a base DN.

1. 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 "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd \
 --filename people-pwp.ldif
Processing ADD request for cn=People Password Policy,dc=example,dc=com
ADD operation successful for DN cn=People Password Policy,dc=example,dc=com
```

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 *Directory Server Developer's Guide* for more information.

2. Check your work:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --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`.

Configuring Password Generation

Password generators are used by OpenDJ 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 OpenDJ directory 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:  eak77qdi
```

The default password policy shown in "To Adjust the Default Password Policy" uses the Random Password Generator, described in "Random Password Generator" in the *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
Property           : Value(s)
-------------------:-------------------------
password-generator : Random Password Generator

$ dsconfig \
 get-password-generator-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --generator-name "Random Password Generator" \
 --property password-generator
 Property               : Value(s)
----------------------:-----------------------------------------------------
enabled                : true
password-character-set : alpha:abcdefghijklmnopqrstuvwxyz, numeric:0123456789
password-format        : "alpha:3,numeric:2,alpha:3"
```

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 OpenDJ 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 \
 --no-prompt

$ ldappasswordmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "u:bjensen"
The LDAP password modify operation was successful
Generated Password:  pld^06:)529HTq$'
```

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.

Configuring Password Storage

Password storage schemes, described in dsconfig create-password-storage-scheme(1) in the *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 offers a variety of both reversible and one-way password storage schemes. Some schemes make it easy to recover the cleartext password, whereas others aim to make it computationally hard to do so:

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

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 in Salted SHA-1. When you add users or import user entries with `userPassword` values in cleartext, OpenDJ 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 Salted SHA-512 by default.

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.`

Scheme	Setting	Description

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 \
 --no-prompt
```

Notice that the change in default password storage scheme does not cause OpenDJ to update any stored password values. By default, OpenDJ only stores a password with the new storage scheme the next time that the password is changed.

OpenDJ prefixes passwords with the scheme used to encode them, which means it is straightforward to see which password storage scheme is in use. After the default password storage scheme is changed to PBKDF2, old user passwords remain encoded with Salted SHA-1:

```
$ 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: {SSHA}Rc3tkAj1qP5zGiRkwDIWDFxrxpGgO8Fwh3aibg==
```

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:O3V6G7y7n7AefOkRGNKQ5ukrMuO5uf+iEQ9ZLg==
```

When you change the password storage scheme for users, realize that the user passwords must change in order for OpenDJ 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 OpenDJ 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: {SSHA}hDgK44F2GhIIZj913b+29Ak7phb9oU3Lz4ogkg==

$ 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" \
 --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:L4dCYqSsNnf47YZ3a6aC8K2E3DChhHHhpcoUzg==
```

Notice that with `deprecated-password-storage-scheme` set appropriately, Kirsten Vaughan's password was hashed again after she authenticated successfully.

Configuring Password Validation

Password validators, described in dsconfig create-password-validator(1) in the *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. OpenDJ directory server provides a variety of password validators:

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


Password Validator                  : Type                 : enabled
------------------------------------:--------------------:--------
Attribute Value                     : attribute-value      : true
Character Set                       : character-set        : 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:

- 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:` would mean 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 \
 --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" \
 --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 the escape characters. In practice it can be easier to enter sequences like that by using `dsconfig` in interactive mode, and letting 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 use in scripts later.

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: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'!"#$%&'()*+,-./:;<=\>?@[\]^_`{|}~', '0123456789', '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.

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Enforce Regular Password Changes" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-1" \
 --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 OpenDJ directory server can write a timestamp value on successful login. For additional information also see "Search: Listing Active Accounts" in the *Directory Server Developer's Guide*:

```
$ ldapmodify \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
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' )

Processing MODIFY request for cn=schema
```

```
MODIFY operation successful for DN cn=schema
```

Next, create the password policy that causes OpenDJ directory server to write the timestamp to the attribute on successful login:

```
$ dsconfig \
 create-password-policy \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Track Last Login Time" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-1" \
 --set password-attribute:userPassword \
 --set last-login-time-attribute:lastLoginTime \
 --set last-login-time-format:"yyyyMMddHH'Z'" \
 --trustAll \
 --no-prompt
```

See also "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", as OpenDJ directory server only employs the new password storage scheme to hash or to encrypt passwords when a password changes:

```
$ dsconfig \
 create-password-policy \
 --port 4444 \
 --hostname opendj.example.com \
 --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-1" \
 --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 OpenDJ directory 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 OpenDJ directory server to accept this new password policy:

```
$ dsconfig \
 create-password-policy \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Lock Idle Accounts" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-1" \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Allow Grace Login" \
 --type password-policy \
 --set default-password-storage-scheme:"Salted SHA-1" \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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-1" \
 --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.

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 directory server supports 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 OpenDJ directory 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. OpenDJ 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*.

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 simple bind attempt failed
Result Code:  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 simple bind attempt failed
Result Code:  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 simple bind attempt failed
Result Code:  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 simple bind attempt failed
Result Code:  49 (Invalid Credentials)
```

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.

For the following examples, the directory admin user, Kirsten Vaughan, has `ds-privilege-name: password-reset`, and the following ACI on `ou=People,dc=example,dc=com`:

```
(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";)
```

To Disable an Account

- Set the account status to disabled with the `manage-account` command:

  ```
  $ manage-account \
   set-account-is-disabled \
   --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 \
   clear-account-is-disabled \
   --port 4444 \
   --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
   --bindPassword bribery \
  ```

```
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --trustAll
Account Is Disabled:  false
```

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 OpenDJ directory server to send a notification about password expiration, that notification gets triggered when the user authenticates during the password expiration warning interval. OpenDJ directory server does not automatically scan entries to send password expiry notifications. OpenDJ directory server does 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set smtp-server:smtp.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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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`

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 used when a user searches the directory
- Limit how long connections can remain idle before they are dropped
- Limit the size of directory server requests

Limiting Search Resources

Well-written directory client applications limit the scope of their searches with filters that narrow the number of results returned. By default, OpenDJ only allows users with appropriate privileges to perform unindexed searches. You can further adjust additional limits on search operations, such as the following:

- The *lookthrough limit* defines the maximum number of candidate entries OpenDJ considers when processing a search.

  The default lookthrough limit, which is set by using the global server property `lookthrough-limit`, is 5000.

  You can override the limit for a particular user by changing the operational attribute, `ds-rlim-lookthrough-limit`, on the user's entry.

- The *size limit* sets the maximum number of entries returned for a search.

  The default size limit, which is set by using the global server property `size-limit`, is 1000.

  You can override the limit for a particular user by changing the operational attribute, `ds-rlim-size-limit`, on the user's entry.

- The *time limit* defines the maximum processing time OpenDJ devotes to a search operation.

  The default time limit, which is set by using the global server property `time-limit`, is 1 minute.

  You can override the limit for a particular user by changing the operational attribute, `ds-rlim-time-limit`, on the user's entry. Times for `ds-rlim-time-limit` are expressed in seconds.

- 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 for a particular user by changing the operational attribute, `ds-rlim-idle-time-limit`, on the user's entry. Times for `ds-rlim-idle-time-limit` are expressed in seconds.

- The maximum number of persistent searches can be set by using the global server property max-psearches.

To Set Search Limits For a User

- Change the user entry to set the limits to override:

  ```
  $ cat limit.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 "cn=Directory Manager" \
   --bindPassword password \
   --filename limit.ldif
  Processing MODIFY request for uid=bjensen,ou=People,dc=example,dc=com
  MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
  ```

  Now when Babs Jensen performs a search returning more than 10 entries, she sees the following message:

  ```
  Result Code:  4 (Size Limit Exceeded)
  Additional Information:  This search operation has sent the maximum of
   10 entries to the client
  ```

To Set Search Limits For a Group

1. Create an LDAP subentry to specify the limits using collective attributes:

   ```
   $ cat grouplim.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 "cn=Directory Manager" \
   ```

```
 --bindPassword password \
 --defaultAdd \
 --filename grouplim.ldif
Processing ADD request for
 cn=Remove Administrator Search Limits,dc=example,dc=com
ADD operation successful for DN
 cn=Remove Administrator Search Limits,dc=example,dc=com
```

2. Check the results:

```
$ ldapsearch --port 1389 --baseDN dc=example,dc=com uid=kvaughan +|grep ds-rlim
ds-rlim-lookthrough-limit: 0
ds-rlim-time-limit: 0
ds-rlim-size-limit: 0
```

Limiting Idle Time

If you have applications that leave connections open for long periods, OpenDJ 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 OpenDJ 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, make sure you 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 does not enforce idle timeout for persistent searches:

```
$ dsconfig \
 set-global-configuration-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set idle-time-limit:24h \
 --trustAll \
 --no-prompt
```

The example shown sets the idle time limit to 24 hours.

Limiting Maximum Request Size

The default maximum request size of 5 MB, set using the advanced connection handler property max-request-size, is sufficient to satisfy most client requests. Yet, there are some cases where you might need to raise the request size limit. For example, if clients add groups with large numbers of members, those add requests can go beyond the 5 MB limit:

```
$ dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "LDAP Connection Handler" \
 --set max-request-size:20mb \
 --trustAll \
 --no-prompt
```

The example shown sets the maximum request size on the LDAP connection handler to 20 MB.

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 *Directory Server Developer's Guide* know 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.

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 OpenDJ's loosely consistent, multi-master data replication. Indeed OpenDJ's 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 provides a unique attribute plugin that you configure by using 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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 bjensen.ldif
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
add: uid
uid: bjensen

$ ldapmodify \
 --defaultAdd \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename bjensen.ldif
Processing MODIFY request for uid=ajensen,ou=People,dc=example,dc=com
MODIFY operation failed
Result Code:  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 might try something like this:

```
$ cat bjensen.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 \
 --defaultAdd \
 --filename bjensen.ldif
Processing ADD request for uid=bjensen,ou=People,dc=example,dc=org
ADD operation failed
Result Code:  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 these alternatives, keep in mind that values must be unique across the attributes and base DNs specified in each plugin configuration. Therefore only group attributes together 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 you create separate unique attribute configurations for `uid `` and `uidNumber`:

   ◦ 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 \
     ```

```
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "UID Unique Attribute" \
 --add type:mobile \
 --trustAll \
 --no-prompt
```

- If you want to create a new plugin configuration, do so as shown in the following example:

```
$ dsconfig \
 create-plugin \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --plugin-name "Unique mobile numbers" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:ou=people,dc=example,dc=com \
 --set type:mobile \
 --trustAll \
 --no-prompt
```

3. Check that the plugin is working correctly:

```
$ cat mobile.ldif
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 828 555 1212

dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 828 555 1212

$ ldapmodify \
 --defaultAdd \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename mobile.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
MODIFY operation failed
```

```
Result Code:  19 (Constraint Violation)
Additional Information:  A unique attribute conflict was detected for
 attribute mobile:  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 \
    --port 4444 \
    --hostname opendj.example.com \
    --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 \
    --port 4444 \
    --hostname opendj.example.com \
    --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 uniq-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 \
 --defaultAdd \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename uniq-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
Result Code:  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.

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 directory server
- 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 conforms 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 OpenDJ when you install, then you must add additional schema definitions to account for the data your applications stored.

OpenDJ comes with many standard schema definitions out of the box. In addition you can update and extend schema definitions while OpenDJ is online. As a result you can add new applications requiring additional data without stopping your directory service.

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.

- *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 OpenDJ 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. OpenDJ directory 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.

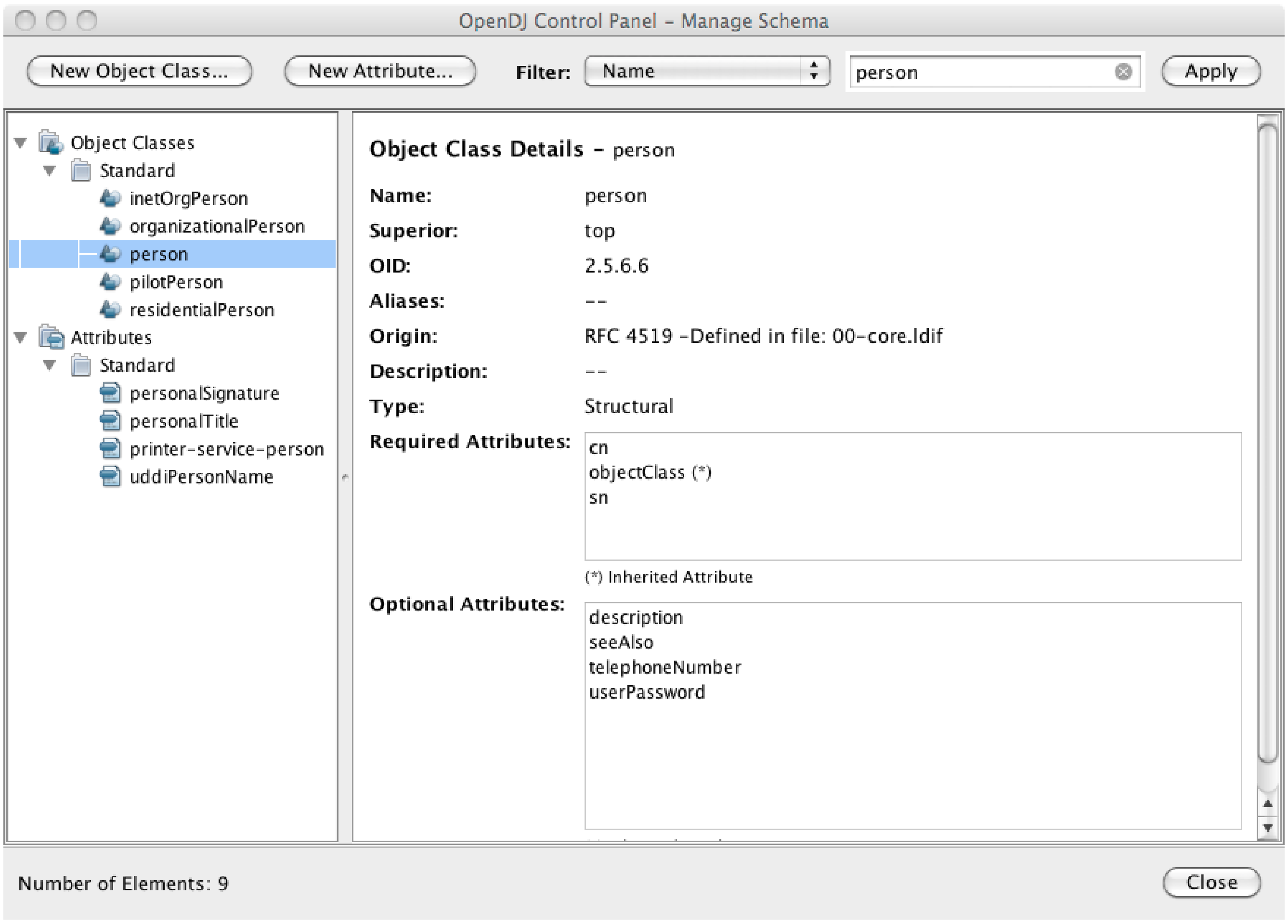
As you can read in examples such as "Search: Listing Active Accounts" in the *Directory Server Developer's Guide*, OpenDJ matching rules enable directory clients to compare other values besides strings, for example.

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.

Updating Directory Schema

OpenDJ directory server is 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 also replicates 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 to get started creating your custom object classes and attribute types.


OpenDJ Control Panel – Manage Schema

Filter: Name — person

Object Classes
- Standard
  - inetOrgPerson
  - organizationalPerson
  - person
  - pilotPerson
  - residentialPerson

Attributes
- Standard
  - personalSignature
  - personalTitle
  - printer-service-person
  - uddiPersonName

Object Class Details – person

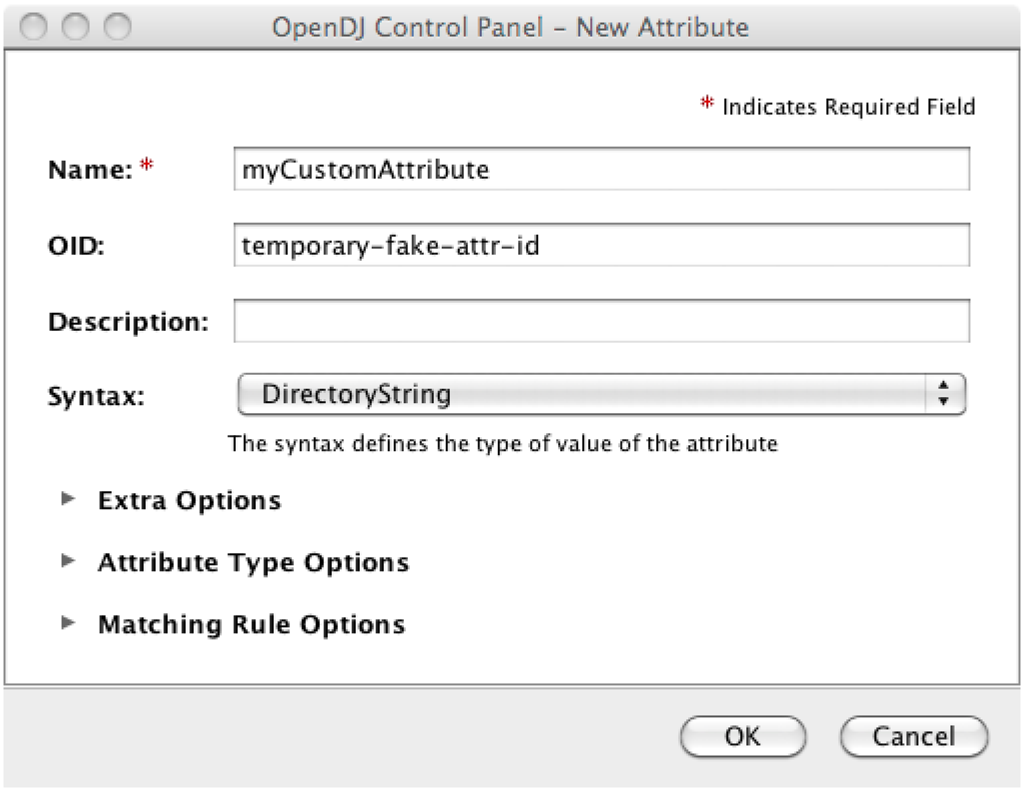
Name:	person
Superior:	top
OID:	2.5.6.6
Aliases:	--
Origin:	RFC 4519 -Defined in file: 00-core.ldif
Description:	--
Type:	Structural
Required Attributes:	cn objectClass (*) sn
Optional Attributes:	description seeAlso telephoneNumber userPassword

(*) Inherited Attribute

Number of Elements: 9

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.

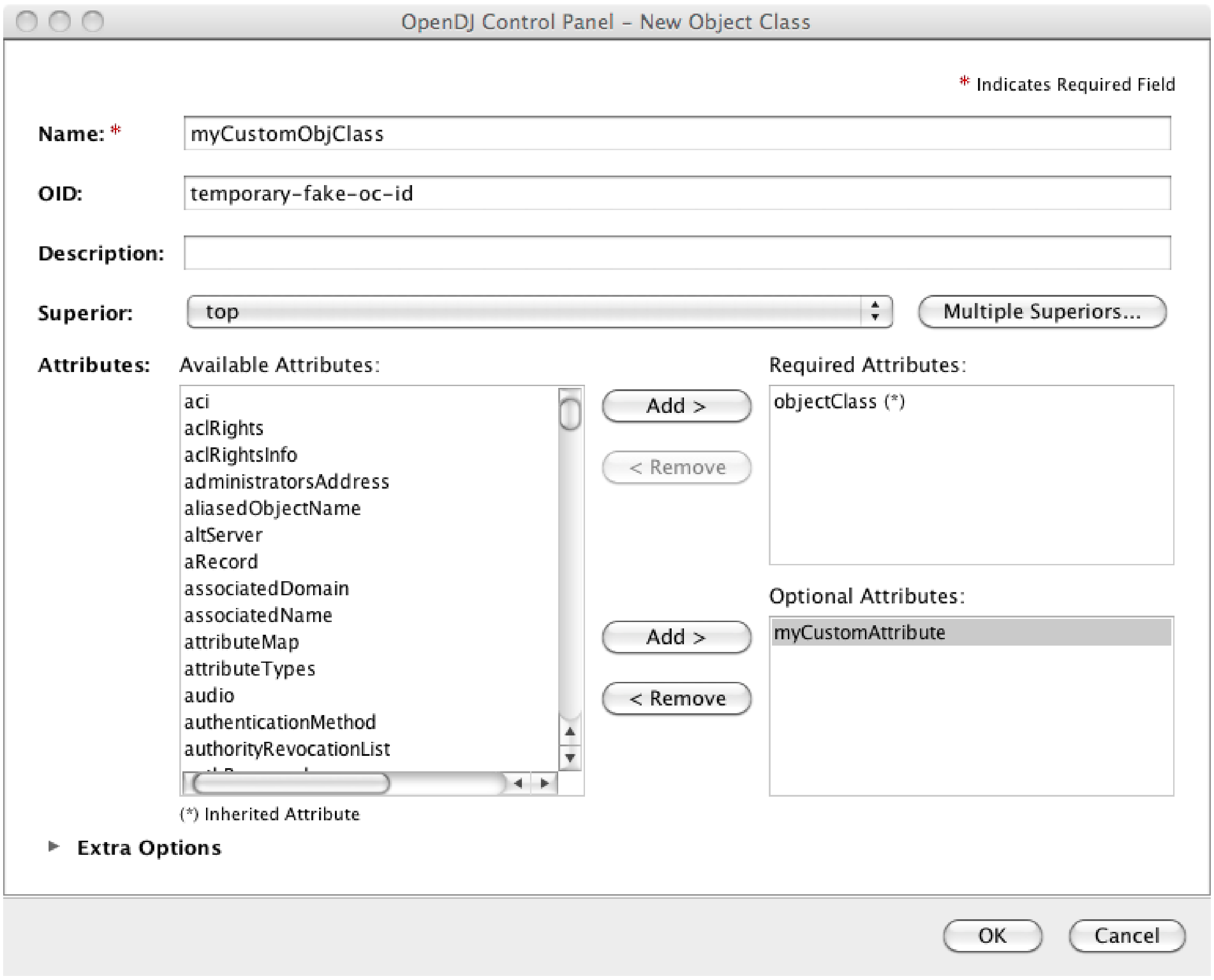

OpenDJ Control Panel – New Attribute

* Indicates Required Field

Name: *	myCustomAttribute
OID:	temporary-fake-attr-id
Description:	
Syntax:	DirectoryString

The syntax defines the type of value of the attribute

Using the New Object Class window, create an auxiliary object class that allows your new custom attribute type. You set the type to Auxiliary under Extra Options.



Name: myCustomObjClass

OID: temporary-fake-oc-id

Description:

Superior: top

Available Attributes: aci, aclRights, aclRightsInfo, administratorsAddress, aliasedObjectName, altServer, aRecord, associatedDomain, associatedName, attributeMap, attributeTypes, audio, authenticationMethod, authorityRevocationList

Required Attributes: objectClass (*)

Optional Attributes: myCustomAttribute

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 reads the schema files in sorted order, reading schema definitions as they occur. If OpenDJ 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:

```
$ cat config/schema/99-user.ldif
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
```

```
attributeTypes: ( temporary-fake-attr-id NAME 'myCustomAttribute' EQUALITY case
 IgnoreMatch ORDERING caseIgnoreOrderingMatch SUBSTR caseIgnoreSubstrings
 Match 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 )
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
modifyTimestamp: 20110620095948Z
```

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 \
 --filename custom-attr.ldif
Processing MODIFY request for uid=bjensen,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com

$ 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, 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 does 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 does 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.

Relaxing Schema Checking to Import Legacy Data

By default, OpenDJ accepts 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
```

Standard Schema Included With OpenDJ Server

OpenDJ directory server provides 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-ietf-boreham-numsubordinates

draft-findlay-ldap-groupofentries

draft-furuseth-ldap-untypedobject

draft-good-ldap-changelog

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 directory 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 directory server configuration.

Working With DIT Structure Rules & Name Forms

This section contains useful information regarding name forms and DIT structure rules.

NOTE

At this time, the OpenDJ Control Panel does not support the management of name forms and DIT structure rules. These schema definition types can only be implemented and managed by way of direct schema file edits (which will necessitate a restart of OpenDJ), *or* through a use of **ldapmodify** against the server's cn=schema context.

Name Forms

From clause 13.1.8 of ITU-T Rec. X.501 and Section 4.1.7.2 of RFC 4512

name form

A name form specifies a permissible RDN for entries of a particular structural object class. A name form identifies a named object class and one or more attribute types to be used for naming (i.e., for the RDN). Name forms are primitive pieces of specification used in the definition of DIT structure rules.

In simplest terms, a name form is a particular schema definition which requires specific RDN syntaxes for use upon entries bearing a specific STRUCTURAL class.

To offer an example of this, consider the following UDDIv3 name form, per the `03-uddiv3.ldif` file included with OpenDJ:

```
nameForms: ( 1.3.6.1.1.10.15.1
   NAME 'uddiBusinessEntityNameForm'
   OC uddiBusinessEntity
   MUST ( uddiBusinessKey )
   X-ORIGIN 'RFC 4403' )
```

This name form states that any entry bearing the STRUCTURAL class `uddiBusinessEntity` MUST ONLY be designated using the `uddiBusinessKey` as the principal RDN attribute type, for example, " `uddiBusinessKey=ABC123` ".

Alternatively, when devising custom name forms, it is possible to enforce the use of specific attribute types within multi-valued RDNs. Consider the following hypothetical name form:

```
nameForms: ( 1.3.6.1.4.1.56521.999.98.15
   NAME 'cnOrgForm'
   OC groupOfUniqueNames
   MUST ( cn $ o ) )
```

This name form states that any entry bearing the STRUCTURAL object class `groupOfUniqueNames` MUST be designated using attribute types `cn` *and* `o` for a qualifying entry bearing a multi-valued RDN, such as " `cn=Auditors+o=Acme Audit Co` ".

Name forms also allow use of MAY clauses. Consider the following hypothetical name form, similar to the above:

```
nameForms: ( 1.3.6.1.4.1.56521.999.98.16
   NAME 'cnOrgAltForm'
   OC groupOfUniqueNames
   MUST cn
```

```
        MAY o )
```

This rule enforces use of the `cn` RDN attribute type the same as before, but while it no longer requires use of `o`, it will not reject it when present. As such, either of the following RDNs are acceptable:

- `cn=Corporate Auditors`
- `cn=Third Party Auditors+o=Acme Audit Co`

But, regardless of the permutations, a name form does little good in practice — unless it is referenced by a DIT structure rule.

DIT Structure Rules

From clause 13.1.6 of ITU-T Rec. X.501 and Section 4.1.7.1 of RFC 4512

DIT structure rule

> *A rule governing the structure of the DIT by specifying a permitted superior to subordinate entry relationship. A structure rule relates a name form, and therefore a structural object class, to superior structure rules. This permits entries of the structural object class identified by the name form to exist in the DIT as subordinates to entries governed by the indicated superior structure rules.*

In short, a DIT structure rule enforces the terms of its prescribed name form. To offer a simple analogy, if a name form presents a law, the DIT structure rule is the public official upholding that law.

Consider this structure rule, per the included `03-uddiv3.ldif` file:

```
    dITStructureRules: ( 1
       NAME 'uddiBusinessEntityStructureRule'
       FORM uddiBusinessEntityNameForm
       X-ORIGIN 'RFC 4403' )
```

This rule employs the `uddiBusinessEntityNameForm` definition, and constrains entries bearing the STRUCTURAL object class of the name form — also known as the `namedObjectClass` — to the RDN attribute type (in this case, `uddiBusinessKey`).

When a DIT structure rule is introduced to the directory schema, it will not be evaluated until an entry is added to the DIT it enforces.

DIT structure rules shall not influence preexisting entries, even if based upon now-illegal STRUCTURAL class and RDN combinations.

Once structure rules have been established, when a new entry is added to, or renamed within the DIT in violation of a structure rule, OpenDJ will return "Object class violation (65)" along with additional contextual information for debugging purposes.

NOTE

As of version 4.8.0, OpenDJ is currently using the result code of "Object class violation (65)" for certain name form related errors, where it should be using "Naming violation (64)".

This issue will be resolved in a future release of the package to avoid introducing breaking changes. Users are advised to update any external scripts or applications which may match the incorrect result code, and take steps to allow recognition of the correct result code in parallel for maximum compatibility.

But when a new entry is successfully added to or renamed within the DIT, a new operational attribute type appears on the entry: governingStructureRule.

From clause 13.1.7 of ITU-T Rec. X.501:

> *Governing structure rule (of an entry): With respect to a particular entry, the single DIT structure rule that applies to the entry. This rule is indicated by the governingStructureRule operational attribute.*

See also Section 3.4.6 of RFC 4512.

In simplest terms, the `governingStructureRule` contains the integer identifier of the DIT structure rule which governs the entry. In the case of the above DIT structure rule, it would appear in LDAP search results as follows:

```
governingStructureRule: 1
```

Instances of this attribute type may be used for diagnostic reasons, or by client applications designed to determine the appropriate RDN syntax to be applied for a new entry, or for an entry being renamed and/or moved, in advance of the request.

DIT structure rules can be configured in such a way that a particular rule extends from, or is subordinate to, another DIT structure rule using the SUP clause.

TIP

A superior DIT structure rule is often referred to as a superior structure rule, per clause 13.1.9 of ITU-T Rec. X.501.

The purpose of the SUP clause is to allow an entry with a particular RDN syntax to reside beneath one of multiple possible choices. For example:

```
    SUP ( 20 21 )
```

In this example, the integer identifiers 20 and 21 indicate that the bearer of this clause will allow entries to reside as subordinates to either of the entries governed by those rules.

Also note that rules can be *recursive* or "self-referencing". This manifests as an instance where a DIT structure rule possesses a SUP clause member that matches its own integer identifier. This is a particularly useful feature because it allows nesting of compliant entries — for example, those bearing the `organizationalUnit` STRUCTURAL class — to exist within superior entries of like-design.

For an example of recursive rules in action, see the `ouStructure` rule (21) in the next section.

DIT Design Under Governance - A Practical Overview

This section will cover the highlights of creating initial DIT content while under the control of easily-understood DIT structure rules enforcing the use of common attribute types within entry RDNs.

The following basic assumptions apply:

- A new `userRoot` backend exists and is identified by the `base-dn` of `dc=example,dc=com`, containing no entries whatsoever, and …
- The eight (8) definitions described have already been saved to `/opt/opendj/config/schema/99-user.ldif` or a similar file, or otherwise added via **ldapmodify**

To begin, let's take a look at the following `nameForms` definitions:

```
    #
      nameForms: ( 1.3.6.1.4.1.56521.999.2.7.1
         NAME 'rootSuffixForm'
         OC domain
         MUST dc )
      #
      nameForms: ( 1.3.6.1.4.1.56521.999.2.7.2
         NAME 'ouForm'
         OC organizationalUnit
         MUST ou )
      #
      nameForms: ( 1.3.6.1.4.1.56521.999.2.7.3
         NAME 'accountForm'
         OC inetOrgPerson
         MUST uid )
      #
      nameForms: ( 1.3.6.1.4.1.56521.999.2.7.4
         NAME 'groupForm'
         OC groupOfNames
         MUST cn )
```

These name forms declare the following mandates:

- Entries bearing the `domain` STRUCTURAL class, MUST utilize `dc` for their respective RDNs
- Entries bearing the `organizationalUnit` STRUCTURAL class, MUST utilize `ou` for their respective RDNs
- Entries bearing the `inetOrgPerson` STRUCTURAL class, MUST utilize `uid` for their respective RDNs
- Entries bearing the `groupOfNames` STRUCTURAL class, MUST utilize `cn` for their respective RDNs

Next, we'll take a look at the new `dITStructureRules` instances, which will bring the above name forms to life:

```
  #
     dITStructureRules: ( 20
                NAME 'rootSuffixStructure'
                FORM rootSuffixForm )
     #
     dITStructureRules: ( 21
                NAME 'ouStructure'
                FORM ouForm
                SUP ( 20 21 ) )
     #
     dITStructureRules: ( 22
                NAME 'accountStructure'
                FORM accountForm
                SUP 21 )
     #
     dITStructureRules: ( 23
                NAME 'groupStructure'
                FORM groupForm
                SUP 21 )
```

From these rules, one can begin to perceive an abstract DIT structure, defined by the incrementing — and hierarchically-significant — integer identifiers, each of which reflect the following respective conditions:

- Given the absence of other entries, the introduction of an entry bearing the `domain` STRUCTURAL class and `dc` RDN attribute signifies the start of the administrative area, or the start of the "chain of enforced rules"

  When added, this entry SHOULD bear a `governingStructureRule` integer identifier of 20

- Given the introduction of an entry, positioned directly subordinate to the root suffix and bearing the `organizationalUnit` STRUCTURAL class and `ou` RDN attribute, the entry is accepted

  When added, this entry SHOULD bear a `governingStructureRule` integer identifier of 21, the subordinate structure rule of its superior structure rule, 20

- Given the introduction of any additional `organizationalUnit` entries, whether descending directly from the root suffix, OR if subordinate to other `organizationalUnit` entries in "nested" fashion, the entry is accepted by rite of structure rule recursion

  When added, this entry SHOULD also bear a `governingStructureRule` integer identifier of 21, as with the previous case

- Given the introduction of an entry, positioned directly subordinate to any `organizationalUnit` entry presently governed by DIT structure rule 21 and bearing the `inetOrgPerson` STRUCTURAL class and `uid` RDN attribute, the entry is accepted

When added, this entry SHOULD bear a governingStructureRule integer identifier of 22

- Given the introduction of an entry, positioned directly subordinate to any organizationalUnit entry presently governed by DIT structure rule 21 and bearing the groupOfNames STRUCTURAL class and cn RDN attribute, the entry is accepted

  When added, this entry SHOULD bear a governingStructureRule integer identifier of 23

Next, we'll be creating the initial portions of the governed DIT using **ldapmodify**, and periodically checking the results with **ldapsearch** along the way.

NOTE In cases where changes are made in this section, the root DN user (cn=Directory Manager) is purposely used. This is simply to demonstrate that no user, regardless of privilege, can "bypass" or otherwise violate DIT structure rules in force.

```
$ ldapmodify -w password \
      -D "cn=Directory Manager" \
      -h opendj.example.com

    dn: dc=example,dc=com
    changetype: add
    objectClass: domain

    Processing ADD request for dc=example,dc=com
    ADD operation successful for DN dc=example,dc=com

    dn: ou=Accounts,dc=example,dc=com
    changetype: add
    objectClass: organizationalUnit

    Processing ADD request for ou=Accounts,dc=example,dc=com
    ADD operation successful for DN ou=Accounts,dc=example,dc=com

    dn: ou=Consultants,ou=Accounts,dc=example,dc=com
    changetype: add
    objectClass: organizationalUnit

    Processing ADD request for ou=Consultants,dc=example,dc=com
    ADD operation successful for DN ou=Consultants,dc=example,dc=com
```

So far, so good. What we've just done is create the initial structure of our DIT, and in doing so we've confirmed the DIT structure rules do not seem to be interfering.

But, let's stop for now and check our work. We want to see the DIT structure rules that are actively governing our entries. To do this, we need only perform a simple anonymous LDAP search:

```
$ ldapsearch -h opendj.example.com \
      -b dc=example,dc=com \
      "(objectClass=*)" \
      governingStructureRule

    dn: dc=example,dc=com
    governingStructureRule: 20

    dn: ou=Accounts,dc=example,dc=com
    governingStructureRule: 21

    dn: ou=Consultants,ou=Accounts,dc=example,dc=com
    governingStructureRule: 21
```

This proves the following:

- Rule 20, the `rootSuffixStructure` definition, represents the start of the structure chain
- Rule 21, the `ouStructure` definition, represents the permitted subordinate naming context below entries governed by the `rootSuffixStructure` rule
- Rule 21, as it supports recursion by nature, allows `organizationalUnit` entries to reside *within* `organizationalUnit` entries, thus allowing categorical organizational structures to exist

Let's see what happens when we attempt to add an entry bearing an unauthorized RDN syntax.

```
$ ldapmodify -w password \
      -D "cn=Directory Manager"\
      -h opendj.example.com

    dn: mail=user@example.com,ou=Consultants,ou=Accounts,dc=example,dc=com
    changetype: add
    objectClass: inetOrgPerson
    cn: User Person
    sn: Person

    Processing ADD request for
    mail=user@example.com,ou=Consultants,ou=Accounts,dc=example,dc=com
    The LDAP modify request failed: 65 (Object Class Violation)
    Additional Information:  Entry
    mail=user@example.com,ou=Consultants,ou=Accounts,dc=example,dc=com violates
    the Directory Server schema configuration because its RDN does not contain
    attribute uid that is required by name form accountForm
```

Good, the DIT structure rule in question seems to work in preventing bogus RDNs. Now let's continue with entries that are expected to work.

```
$ ldapmodify -w password \
      -D "cn=Directory Manager" \
      -h opendj.example.com

    dn: uid=userPerson,ou=Consultants,ou=Accounts,dc=example,dc=com
    changetype: add
    objectClass: inetOrgPerson
    sn: Person
    cn: User Person

    Processing ADD request for
uid=userPerson,ou=Consultants,ou=Accounts,dc=example,dc=com
    ADD operation successful for DN
uid=userPerson,ou=Consultants,ou=Accounts,dc=example,dc=com

    dn: ou=Groups,dc=example,dc=com
    changetype: add
    objectClass: organizationalUnit

    Processing ADD request for ou=Groups,dc=example,dc=com
    ADD operation successful for DN ou=Groups,dc=example,dc=com

    dn: ou=Corporate,ou=Groups,dc=example,dc=com
    changetype: add
    objectClass: organizationalUnit

    Processing ADD request for ou=Corporate,ou=Groups,dc=example,dc=com
    ADD operation successful for DN ou=Corporate,ou=Groups,dc=example,dc=com

    dn: ou=Infrastructure,ou=Groups,dc=example,dc=com
    changetype: add
    objectClass: organizationalUnit

    Processing ADD request for ou=Infrastructure,ou=Groups,dc=example,dc=com
    ADD operation successful for DN ou=Infrastructure,ou=Groups,dc=example,dc=com

    dn: cn=Abuse Mail,ou=Infrastructure,ou=Groups,dc=example,dc=com
    changetype: add
    objectClass: groupOfNames

    Processing ADD request for cn=Abuse
Mail,ou=Infrastructure,ou=Groups,dc=example,dc=com
    ADD operation successful for DN cn=Abuse
Mail,ou=Infrastructure,ou=Groups,dc=example,dc=com
```

Again, let’s check our work (omitting the contents of the previous LDAP search):

```
$ ldapsearch -h opendj.example.com \
      -b dc=example,dc=com \
      "(objectClass=*)" \
      governingStructureRule

    dn: uid=userPerson,ou=Consultants,ou=Accounts,dc=example,dc=com
    governingStructureRule: 22

    dn: ou=Groups,dc=example,dc=com
```

```
governingStructureRule: 21

dn: ou=Corporate,ou=Groups,dc=example,dc=com
governingStructureRule: 21

dn: ou=Infrastructure,ou=Groups,dc=example,dc=com
governingStructureRule: 21

dn: cn=Abuse Mail,ou=Infrastructure,ou=Groups,dc=example,dc=com
governingStructureRule: 23
```

So, what did we learn?

- `ouStructure` rule 21 continues to allow recursive `organizationalUnit` entries, so long as they ultimately extend from the `rootSuffixStructure` superior structure (ancestor) rule 20, *or* another such entry governed by rule 21
- `accountStructure` rule 22 is correctly governing entries bearing the `inetOrgPerson` STRUCTURAL class found within an `organizationalUnit` entry (superior structure rule 21)
- `groupStructure` rule 23 is correctly governing entries bearing the `groupOfNames` STRUCTURAL class found within an `organizationalUnit` entry (superior structure rule 21)

DIT structure rules are extremely powerful. When properly planned and implemented, they can greatly aid in the formation of clean and orderly directory structures without the need for additional ACIs.

Considerations Relating To The Implementation Of DIT Structure Rules In An Established DIT

Because DIT structure rules do not influence preexisting entries, even those in violation of those rules, this presents a potential pain-point regarding the restoration of content that (in some way) predates the incorporation of those DIT structure rules. This situation may apply following a disaster-triggered reload of data, or when using this data to "seed" a new DSA being built in the topology.

If DIT structure rules are already applied to the DSA in question, but data has NOT yet been loaded, the DIT structure rules in question will consider ANY data to be "new" regardless of its true chronological age.

If violations are perceived, this will result in errors during the incorporation of that data. This can be confusing to administrators if that same data exists as expected on other DSAs — even those with effectively identical configurations.

When introducing DIT structure rules to an established (preexisting) DIT, it is strongly recommended that separate load-tests be conducted on a disposable system or virtual image that is under the governance of all planned DIT structure rules. This will allow accurate simulation of new in-topology server builds, or rebuilds of preexisting servers that have suffered a malfunction of some kind, or have been rebuilt due to upgrade or other reasons.

Considerations For Collective Attribute Subentries

Directories which utilize both DIT structure rules as well as collective attribute subentries, per RFC 3672, will require specific adjustments to allow these two features to cooperate.

Clause 14.2.2 of ITU-T Rec. X.501 defines the `subentryNameForm` ASN.1 definition, which is intended solely to enforce a singular naming convention for subentries:

```
subentryNameForm NAME-FORM ::= {
  NAMES subentry
  WITH ATTRIBUTES {commonName}
  ID id-nf-subentryNameForm }
```

The same subsection also states that "No other name form shall be used for subentries". In other words, according to this standard, this is the ONLY permitted name form for subentries.

As such, the OpenDJ package conveniently includes the equivalent LDAP name form definition within the subschema subentry for users to leverage:

```
nameForms: ( 2.5.15.16
   NAME 'subentryNameForm'
   DESC 'X.501, cl. 14.2.2: the Subentry name form'
   OC subentry
   MUST cn )
```

As a result, users are only expected to implement the DIT structure rule meant to reference this name form.

Consider the following fictional structure rule definition, which contains two distinct "placeholders" the user needs to populate:

```
dITStructureRules: ( <structure rule ID>
   NAME 'subentryStructure'
   FORM subentryNameForm
   SUP ( <superior structure rule ID(s)> ) )
```

As with any new structure rule, the user will need to first assign a unique integer identifier to replace the `<structure rule ID>` placeholder. For the purposes of this example, let’s assume "88" is chosen.

Next, all superior structure rule integer identifiers under which a `subentry` *COULD* be created MUST be referenced. For the purposes of this example, let’s assume the superior structure rule identifiers chosen are "15" and "16", meaning that `subentry` instances are permitted as subordinate entries to those parent entries which currently bear a matching `governingStructureRule` integer identifier.

As such, the final structure rule would appear as:

```
    dITStructureRules: ( 88
       NAME 'subentryStructure'
       FORM subentryNameForm
       SUP ( 15 16 ) )
```

Because subentries themselves do not allow for subordinate entries, we need not worry about rule recursion in this instance.

Again, this procedure is only necessary in directories which utilize DIT structure rules *and* collective attribute subentries.

ACIs Vs. DIT Structure Rules

Some LDAP implementations on the market today offer no support for DIT structure rules. A common workaround for this is the use of ACIs to enforce specific naming conventions for entries. While OpenDJ supports this technique just the same, there are potential caveats.

Use of ACIs to enforce such rules can be bypassed by users with sufficient access privileges. DIT structure rules, on the other hand, are defined in the schema, which conceptually exists at a lower and more fundamental level than ACIs. As such, no user can bypass a DIT structure rule using conventional means — not even the root DN.

There is also the classic argument that use of ACIs to effect "behavioral changes" in this manner is contrary to the very intent of ACIs. Because DIT structure rules are essentially immutable and do not discriminate the origin of any request, they resemble configuration directives in practice more so than an expression of privilege.

The argument against ACIs in this context gains additional momentum when one considers the innate risk of altering ACIs for any reason, as even the slightest misstep can deny critical functionality or, worse, expose data.

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

About Pass-through Authentication

You use *pass-through authentication* (PTA) when the credentials for authenticating are stored in a remote directory service instead of OpenDJ. In effect OpenDJ redirects the bind operation against a remote LDAP server. The method OpenDJ uses to redirect the bind depends on the mapping from the user entry in OpenDJ to the corresponding user entry in the remote directory. OpenDJ provides you several choices to set up the mapping:

- When both the local entry in OpenDJ and the remote entry in the other server have the same DN, you do not have to set up the mapping. By default, OpenDJ redirects the bind with the original DN and password from the client application.
- When the local entry in OpenDJ has been provisioned with an attribute holding the DN of the remote entry, you can specify which attribute holds the DN, and OpenDJ 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. OpenDJ 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.

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 OpenDJ to user entries in the remote directory.

To Set Up SSL Communication For Testing

When performing PTA, you protect communications between OpenDJ and the server providing authentication. If you test using SSL 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 OpenDJ keystore.

1. Export the server certificate from the authentication server.

   How you perform this step depends on the authentication directory server. With OpenDJ, you can export the certificate as shown here:

   ```
   $ cd /path/to/PTA-Server/config
   $ keytool \
    -exportcert \
    -rfc \
    -alias server-cert \
    -keystore keystore \
    -storepass `cat keystore.pin` \
    > /tmp/pta-srv-cert.pem
   ```

2. Make note of the host name used in the certificate.

   You use the host name when configuring the SSL connection. With OpenDJ, you can view the certificate details as shown here:

   ```
   $ keytool \
    -list \
    -v \
    -alias server-cert \
    -keystore keystore \
    -storepass `cat keystore.pin`
   Alias name: server-cert
   Creation date: Sep 12, 2011
   Entry type: PrivateKeyEntry
   Certificate chain length: 1
   Certificate[1]:
   Owner: CN=pta-server.example.com, O=OpenDJ Self-Signed Certificate
   Issuer: CN=pta-server.example.com, O=OpenDJ Self-Signed Certificate
   Serial number: 4e6dc429
   Valid from: Mon Sep 12 10:34:49 CEST 2011 until: Wed Sep 11 10:34:49 CEST 2013
   Certificate fingerprints:
     MD5:  B6:EE:1C:A0:71:12:EF:6F:21:24:B9:50:EF:8B:4E:6A
     SHA1: 7E:A1:C9:07:D2:86:56:31:24:14:F7:07:A8:6B:3E:A1:39:63:F4:0E
     Signature algorithm name: SHA1withRSA
     Version: 3
   ```

3. Import the authentication server certificate into OpenDJ's keystore:

   ```
   $ cd /path/to/opendj/config
   $ keytool \
    -importcert \
    -alias pta-cert \
    -keystore truststore \
    -storepass `cat keystore.pin` \
   ```

```
 -file /tmp/pta-srv-cert.pem
Owner: CN=pta-server.example.com, O=OpenDJ Self-Signed Certificate
Issuer: CN=pta-server.example.com, O=OpenDJ Self-Signed Certificate
Serial number: 4e6dc429
Valid from: Mon Sep 12 10:34:49 CEST 2011 until: Wed Sep 11 10:34:49 CEST 2013
Certificate fingerprints:
  MD5:  B6:EE:1C:A0:71:12:EF:6F:21:24:B9:50:EF:8B:4E:6A
  SHA1: 7E:A1:C9:07:D2:86:56:31:24:14:F7:07:A8:6B:3E:A1:39:63:F4:0E
  Signature algorithm name: SHA1withRSA
  Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

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 \
    --port 4444 \
    --hostname opendj.example.com \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --type ldap-pass-through \
    --policy-name "PTA Policy" \
    --set primary-remote-ldap-server:pta-server.example.com:636 \
    --set mapped-attribute:uid \
    --set mapped-search-base-dn:"dc=PTA Server,dc=com" \
    --set mapping-policy:mapped-search \
    --set use-ssl:true \
    --set trust-manager-provider:JKS \
    --trustAll \
    --no-prompt
   ```

   The policy shown here maps identities with this this password policy to identities under `dc=PTA Server,dc=com`. Users must have the same `uid` values on both servers. The policy here also uses SSL between OpenDJ and the authentication server.

2. Check that your policy has been added to the list:

   ```
   $ dsconfig \
    list-password-policies \
    --port 4444 \
    --hostname opendj.example.com \
   ```

```
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --property use-ssl

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 OpenDJ authenticate through to 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
```

OpenDJ 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, OpenDJ has to perform a search for the user in Active Directory using the `cn` value 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 OpenDJ uses to bind to Active Directory to be able to search.

Finally, before setting up the PTA policy, make sure OpenDJ 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 OpenDJ.
3. Import the server certificate into OpenDJ's keystore:

```
$ cd /path/to/opendj/config
$ keytool \
 -importcert \
 -alias ad-cert \
 -keystore truststore \
 -storepass `cat keystore.pin` \
 -file ~/Downloads/windows.cer
Owner: CN=internal-ACTIVEDIRECTORY-CA, DC=internal, DC=forgerock, DC=com
Issuer: CN=internal-ACTIVEDIRECTORY-CA, DC=internal, DC=forgerock, DC=com
Serial number: 587465257200a7b14a6976cb47916b32
Valid from: Tue Sep 20 11:14:24 CEST 2011 until: Tue Sep 20 11:24:23 CEST 2016
Certificate fingerprints:
  MD5:  A3:D6:F1:8D:0D:F9:9C:76:00:BC:84:8A:14:55:28:38
  SHA1: 0F:BD:45:E6:21:DF:BD:6A:CA:8A:7C:1D:F9:DA:A1:8E:8A:0D:A4:BF
  Signature algorithm name: SHA1withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=false
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: A3 3E C0 E3 B2 76 15 DC   97 D0 B3 C0 2E 77 8A 11  .>...v.......w..
0010: 24 62 70 0A                                        $bp.
]
]

#4: ObjectId: 1.3.6.1.4.1.311.21.1 Criticality=false

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

At this point OpenDJ can connect to Active Directory over SSL.

4. Set up an authentication policy for OpenDJ users to authenticate to Active Directory:

```
$ dsconfig \
 create-password-policy \
 --port 4444 \
 --hostname opendj.example.com \
 --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:JKS \
 --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.

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`:

```
$ ldapsearch \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 uid=user.0 \
 pwdPolicySubentry
dn: uid=user.0,ou=People,dc=example,dc=com
```

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 on OpenDJ is as shown. Notice that the user has no password set. The user's password on the authentication server is `password`:

```
dn: uid=user.0,ou=People,dc=example,dc=com
cn: Aaccf Amar
```

```
description: This is the description for Aaccf Amar.
employeeNumber: 0
givenName: Aaccf
homePhone: +1 225 216 5900
initials: ASA
l: Panama City
mail: user.0@maildomain.net
mobile: +1 010 154 3228
objectClass: person
objectClass: inetorgperson
objectClass: organizationalperson
objectClass: top
pager: +1 779 041 6341
postalAddress: Aaccf Amar$01251 Chestnut Street$Panama City, DE  50369
postalCode: 50369
sn: Amar
st: DE
street: 01251 Chestnut Street
telephoneNumber: +1 685 622 6202
uid: user.0
```

This user's entry on the authentication server also has `uid=user.0`, and the pass-through authentication policy performs the mapping to find the user entry in the authentication server.

1. Prevent users from changing their own password policies:

```
$ cat protect-pta.ldif
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///uid=*,ou=People,dc=example,dc=com")(targetattr =
 "ds-pwp-password-policy-dn")(version 3.0;acl "Cannot choose own pass
 word policy";deny (write)(userdn = "ldap:///self");)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename protect-pta.ldif
Processing MODIFY request for ou=People,dc=example,dc=com
MODIFY operation successful for DN ou=People,dc=example,dc=com
```

2. Update the user's `ds-pwp-password-policy-dn` attribute:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password
dn: uid=user.0,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

Processing MODIFY request for uid=user.0,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=user.0,ou=People,dc=example,dc=com
```

3. Check that the user can authenticate through to the authentication server:

```
$ ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 --bindDN uid=user.0,ou=People,dc=example,dc=com \
 --bindPassword password \
 uid=user.0 \
 cn sn
dn: uid=user.0,ou=People,dc=example,dc=com
cn: Aaccf Amar
sn: Amar
```

To Assign a Pass-Through Authentication Policy To a Group

Examples in the following steps use the PTA policy as defined above. Kirsten Vaughan’s entry has been reproduced on the authentication server under `dc=PTA Server,dc=com`.

1. Create a subentry to assign a collective attribute that sets the `ds-pwp-password-policy-dn` attribute for group members' entries:

```
$ cat pta-coll.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 \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd \
 --filename pta-coll.ldif
Processing ADD request for cn=PTA Policy for Dir Admins,dc=example,dc=com
```

```
ADD operation successful for DN cn=PTA Policy for Dir Admins,dc=example,dc=com
```

2. Check that OpenDJ has applied the policy.

   a. Make sure you can bind as the user on the authentication server:

   ```
   $ ldapsearch \
    --port 2389 \
    --bindDN "uid=kvaughan,ou=People,dc=PTA Server,dc=com" \
    --bindPassword password \
    --baseDN "dc=PTA Server,dc=com" \
    uid=kvaughan
   dn: uid=kvaughan,ou=People,dc=PTA Server,dc=com
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   objectClass: top
   givenName: Kirsten
   uid: kvaughan
   cn: Kirsten Vaughan
   sn: Vaughan
   userPassword: {SSHA}x1BdtrJyRTw63kBSJFDvgvd4guzk66CV8L+t8w==
   ou: People
   mail: jvaughan@example.com
   ```

   b. Check that the user can authenticate through to the authentication server from OpenDJ directory server:

   ```
   $ ldapsearch \
    --port 1389 \
    --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
    --bindPassword password \
    --baseDN dc=example,dc=com \
    uid=kvaughan \
    cn sn
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   cn: Kirsten Vaughan
   sn: Vaughan
   ```

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 OpenDJ directory server
- Set up the OpenDJ directory sever 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 OpenDJ for storing Samba accounts. See the Samba documentation for your platform for instructions on how to configure an LDAP directory server such as OpenDJ as a Samba passdb backend.

The rest of this chapter focuses on how you keep passwords in sync when using OpenDJ for Samba account storage.

When you store Samba accounts in OpenDJ, 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 \
 --defaultAdd \
 --filename samba.ldif
Processing ADD request for uid=samba-admin,ou=Special Users,dc=example,dc=com
ADD operation successful for DN uid=samba-admin,ou=Special Users,
 dc=example,dc=com
```

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) groupdn ="ldap:///uid=samba-user,ou=
 Special Users,dc=example,dc=com";)

$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename samba-rights.ldif
Processing MODIFY request for uid=samba-admin,ou=Special
Users,dc=example,dc=com
MODIFY operation successful for DN
 uid=samba-admin,ou=Special Users,dc=example,dc=com
Processing MODIFY request for dc=example,dc=com
MODIFY operation successful for DN dc=example,dc=com
```

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:

```
$ dconfig \
 create-plugin \
 --port 4444 \
 --hostname opendj.example.com \
 --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:

```
$ dconfig \
 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 \
 --no-prompt \
 --trustAll

$ dconfig \
 create-debug-target \
 --port 4444 \
 --hostname opendj.example.com \
 --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
```

Monitoring, Logging, and Alerts

This chapter covers OpenDJ monitoring capabilities. In this chapter you will learn to:

- Access monitoring information over LDAP, over SNMP, and though use of JMX.
- Monitor directory server status, including the status of directory server tasks
- Configure directory server 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.

LDAP-Based Monitoring

OpenDJ exposes monitoring information over LDAP under the entry `cn=monitor`. Many different types of information are exposed. The following example shows monitoring information about the `userRoot` backend holding Example.com data:

Interface stability: Evolving (See "Product Interface Stability" in the *Reference*)

```
$ ldapsearch --port 1389 --baseDN cn=monitor "(cn=userRoot backend)"
dn: cn=userRoot backend,cn=Disk Space Monitor,cn=monitor
disk-state: normal
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
disk-dir: /path/to/opendj/db/userRoot
disk-free: 343039315968
cn: userRoot backend

dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
cn: userRoot Backend
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: FALSE
ds-backend-entry-count: 176
ds-base-dn-entry-count: 176 dc=example,dc=com
ds-backend-writability-mode: enabled
```

You can set global ACIs on the Access Control Handler if you want to limit read access under `cn=monitor`.

SNMP-Based Monitoring

OpenDJ lets you monitor the server over SNMP with support for 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 separately. OpenDJ directory server that you download from GitHub is built with OpenDMK, but due to licensing OpenDMK is not part of OpenDJ. SNMP is therefore not enabled by default.

To run the OpenDMK installer, use the self-extracting .jar:

```
$ java -jar ~/Downloads/opendmk-1.0-b02-*.jar
$ 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`.

Once you have installed OpenDMK, you can set up a connection handler for SNMP by enabling the connection handler, and pointing OpenDJ to your installation of the OpenDMK `jdmkrt.jar` library:

```
$ dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --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. Therefore if you install as a normal user, you might want to change the listen and trap ports:

```
$ dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --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 port number changes into account.

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

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

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

Use a command such as `snmpwalk` to check that the SNMP listen port works:

```
$ snmpwalk -v 2c -c OpenDJ@OpenDJ localhost:11161
SNMPv2-SMI::mib-2.66.1.1.1.1 = STRING: "OpenDJ 4.9.2..."
SNMPv2-SMI::mib-2.66.1.1.2.1 = STRING: "/path/to/opendj"
...
```

JMX-Based Monitoring

OpenDJ provides JMX-based monitoring. A number of tools support JMX, including `jconsole` and `jvisualvm`, which are 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 "Product Interface Stability" in the *Reference*)

Configure the server to activate JMX access. The following example uses the reserved port number, 1689:

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

Add appropriate privileges to access JMX monitoring information. By default, no users have privileges to access the JMX connection. The following commands create a user with JMX privileges, who can authenticate over an insecure connection:

```
$ bin/dsconfig
   create-password-policy
   --policy-name "Allow insecure authentication"
   --type password-policy
   --set default-password-storage-scheme:PBKDF2-HMAC-SHA256
   --set password-attribute:userPassword
   --trustAll --no-prompt
   --hostname opendj.example.com
   --port 4444
   --bindDN "cn=Directory Manager"
   --bindPassword passwordt
```

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword password
   dn: uid=JMX Monitor,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   cn: JMX Monitor
   sn: User
   uid: JMX Monitor
   userPassword: password
   ds-privilege-name: monitor-read
   ds-privilege-name: jmx-notify
   ds-privilege-name: jmx-read
   ds-privilege-name: jmx-write
   ds-pwp-password-policy-dn: cn=Allow insecure authentication,cn=Password
Policies,cn=config

   Processing ADD request for uid=JMX Monitor,dc=example,dc=com
   ADD operation successful for DN uid=JMX Monitor,dc=example,dc=com
   ^C
```

Connect remotely.

```
$ jconsole &
```

Remote process

`service:jmx:rmi:///jndi/rmi://localhost:1689/org.opends.server.protocols.jmx.client-unknown`

Username

`uid=JMX Monitor,dc=example,dc=com`

Password

`password`

Connect

Insecure connection

Server Operation and Tasks

OpenDJ comes with two 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

          --- Server Status ---
Server Run Status:        Started
Open Connections:         1

          --- Server Details ---
Host Name:                localhost
Administrative Users:     cn=Directory Manager
Installation Path:        /path/to/opendj
Version:                  OpenDJ 4.9.2
Java Version:             version
Administration Connector: Port 4444 (LDAPS)

          --- Connection Handlers ---
Address:Port : Protocol : State
-------------:----------:---------
--           : LDIF     : Disabled
0.0.0.0:636  : LDAPS    : Disabled
0.0.0.0:1389 : LDAP     : Enabled
0.0.0.0:1689 : JMX      : Disabled
```

```
          --- Data Sources ---
Base DN:     dc=example,dc=com
Backend ID:  userRoot
Entries:     163
Replication: Disabled
```

The `manage-tasks` command connects over 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

ID                         Type    Status
--------------------------------------------------------
example                    Backup  Recurring
example-2011062303000000   Backup  Waiting on start time
```

Server Logs

By default OpenDJ stores access and errors logs, and a server process ID file under the `logs/` directory. For the replication service, OpenDJ also keeps a replication log there. You can also configure a debug log. You can also configure policies about how logs are rotated, and how they are retained. You configure logging using the `dsconfig` command.

Each log depends on a *log publisher*, whose type corresponds to the type of log. OpenDJ provides a number of file-based log publishers out of the box, and supports the Open Identity Platform common audit event framework, sometimes referred to as Common Audit. The ForgeRock common audit event framework provides log handlers for publishing to CSV files, relational databases, and the UNIX system log (Syslog) as described in "Common ForgeRock Access Logs". The framework makes it possible to plug in additional handlers as well.

Access Logs

The *access log* traces the operations the server processes including timestamps, connection information, and information about the operation itself. The access log can grow quickly, as each client request results in at least one new log message.

The following access log excerpt shows a search operation from the local host, with the first three lines wrapped for readability:

```
[21/Jun/2011:08:01:53 +0200] CONNECT conn=4 from=127.0.0.1:49708
```

```
 to=127.0.0.1:1389 protocol=LDAP
[21/Jun/2011:08:01:53 +0200] SEARCH REQ conn=4 op=0 msgID=1
 base="dc=example,dc=com" scope=wholeSubtree filter="(uid=bjensen)" attrs="ALL"
[21/Jun/2011:08:01:53 +0200] SEARCH RES conn=4 op=0 msgID=1
 result=0 nentries=1 etime=3
[21/Jun/2011:08:01:53 +0200] UNBIND REQ conn=4 op=1 msgID=2
[21/Jun/2011:08:01:53 +0200] DISCONNECT conn=4 reason="Client Unbind"
```

Notice that by default OpenDJ directory server logs a message for the search request, and a message for the search response.[1] The server also logs request and response messages for other operations that have responses, such as bind and modify operations. The server does not log response messages for all operations, as some operations, such as persistent searches, abandon operations, unbind operations, and abandoned operations, do not have responses. In the preceding excerpt, notice that the log message for the unbind request is followed by a log message for the disconnection.

Common Open Identity Platform Access Logs

In addition to the default file-based access log formats, OpenDJ directory server supports the Open Identity Platform common audit event framework. OpenDJ uses the framework to write access logs in formats that are compatible with all products using the framework. The framework uses transaction IDs that make it easy to correlate requests as they traverse the platform. This makes it easier to monitor activity and to enrich reports.

Interface stability: Evolving (See "Product Interface Stability" in the *Reference*)

The Open Identity Platform common audit event framework is built around audit event handlers. Audit event handlers can encapsulate their own configurations. Audit event handlers are the same in each product in the Open Identity Platform. As a result, you can plug in custom handlers that comply with the framework without having to upgrade OpenDJ directory server. The Open Identity Platform common audit event framework includes handlers for logging audit event messages to local files and facilities, as well as to remote systems. Handlers for the following are supported:

- CSV files, with support for tamper-evident logs.

  OpenDJ supports LDAP and HTTP CSV access logs, which you must configure in order to use.

- Elasticsearch server.

  You configure the Elasticsearch handler as an external log publisher that logs access messages to Elasticsearch.

- Relational database using JDBC.

  You configure the JDBC handler as an external log publisher that logs access messages to a relational database.

- The UNIX system log facility.

  Although it is rarely used for access events, you can configure the Syslog handler as an external log publisher that logs access messages to the UNIX Syslog facility.

The Open Identity Platform common audit event framework supports a variety of audit event topics. OpenDJ currently supports handling for access events, which are system boundary events such as the initial request and final response to that request. In other words, the implementation in OpenDJ is focused only on access logging. Based on the connection handler for the request, OpenDJ divides access events into `ldap-access` events and `http-access` events. To enable common audit-based logging, follow one of these procedures:

- "To Enable LDAP CSV Access Logs"
- "To Enable HTTP CSV Access Logs"
- "To Enable External LDAP or HTTP Access Logging"

To Enable LDAP CSV Access Logs

After you complete the following steps, OpenDJ directory server records LDAP access event messages in files named like `logs/ldap-access.csv`:

1. (Optional) If you trust transaction IDs sent by client applications, and want monitoring and reporting systems consuming the logs to allow correlation of requests as they traverse multiple servers, update the global server configuration as described in "To Trust Transaction IDs".

2. Create an enabled CSV File Access Log Publisher with optional rotation and retention policies as in the following example:

```
$ dsconfig \
 create-log-publisher \
 --port 4444 \
 --hostname opendj.example.com \
 --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
```

You can view the log publisher properties to check your work as in the following example:

```
$ dsconfig \
 get-log-publisher-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
 --trustAll \
 --no-prompt
Property           : Value(s)
-------------------:-----------------------------------------------------------
csv-delimiter-char : ","
enabled            : true
filtering-policy   : no-filtering
key-store-file     : -
key-store-pin-file : -
log-control-oids   : false
log-directory      : logs
retention-policy   : File Count Retention Policy
rotation-policy    : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                   : Policy
tamper-evident     : false
```

Notice that 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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.

Make certain that you test the performance impact of tamper-evident logging with realistic access patterns for your deployment before enabling the feature in production.

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.

After you complete the following steps, OpenDJ directory server records HTTP access event messages in files named like `logs/http-access.csv`:

1. (Optional) If you trust transaction IDs sent by client applications, and want monitoring and reporting systems consuming the logs to allow correlation of requests as they traverse multiple servers, update the global server configuration as described in "To Trust Transaction IDs".
2. Create an enabled CSV File HTTP Access Log Publisher with optional rotation and retention policies as in the following example:

```
$ dsconfig \
 create-log-publisher \
 --port 4444 \
 --hostname opendj.example.com \
 --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
```

   You can view the log publisher properties to check your work as in the following example:

```
$ dsconfig \
 get-log-publisher-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access Logger" \
 --trustAll \
 --no-prompt
Property           : Value(s)
-------------------:-------------------------------------------------------------
csv-delimiter-char : ","
enabled            : true
key-store-file     : -
key-store-pin-file : -
log-directory      : logs
```

```
retention-policy   : File Count Retention Policy
rotation-policy    : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                   : Policy
tamper-evident     : false
```

Notice that 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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.

   Make certain that you test the performance impact of tamper-evident logging with realistic access patterns for your deployment before enabling the feature in production.

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:

```
$ echo password > /path/to/opendj/config/audit-keystore.pin
$ chmod 400 /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 `cat /path/to/opendj/config/audit-keystore.pin` \
 -keypass `cat /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 `cat /path/to/opendj/config/audit-keystore.pin` \
 -keypass `cat /path/to/opendj/config/audit-keystore.pin`
```

4. Verify the contents of the keystore:

```
$ keytool \
 -list \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass `cat /path/to/opendj/config/audit-keystore.pin`

Keystore type: JCEKS
Keystore provider: SunJCE
```

```
Your keystore contains 2 entries

signature, Nov 27, 2015, PrivateKeyEntry,
Certificate fingerprint (SHA1): 4D:CF:CC:29:...:8B:6E:68:D1
password, Nov 27, 2015, SecretKeyEntry,
```

To Enable External LDAP or HTTP Access Logging

External LDAP or HTTP access event logging lets you use an Elasticsearch handler to log to an Elasticsearch server, a JDBC handler to log to a relational database, a Syslog handler to log to the UNIX Syslog facility, or a custom handler to consume the events in some other way. The configuration depends on the handler, and is provided as a JSON file that corresponds to the handler.

Follow these steps:

1. (Optional) If you trust transaction IDs sent by client applications, and want monitoring and reporting systems consuming the logs to allow correlation of requests as they traverse multiple servers, update the global server configuration as described in "To Trust Transaction IDs".
2. If necessary, prepare the data store:

   - For an Elasticsearch server, create a mapping in the index for the messages.

     See "Using an Elasticsearch Audit Log Handler".

   - For the relational database that the JDBC handler connects to, create the necessary schema and tables.

     See the examples in the `db` directory inside the `opendj/lib/forgerock-audit-handler-jdbc.jar` file.

   The columns and fields of the audit event messages correspond to the fields in the logs generated by the CSV audit handler.

3. Create the JSON configuration file for the external handler, and copy it to the `config` directory for the OpenDJ directory server.

   For details, see "JDBC Audit Event Handler Configuration" and "Syslog Audit Event Handler Configuration".

4. (Optional) For LDAP access logging, create an External Access Log Publisher

   The following example creates a JDBC LDAP access log publisher:

```
$ dsconfig \
 create-log-publisher \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "JDBC LDAP Access Log Publisher" \
 --type external-access \
 --set enabled:true \
 --set config-file:config/jdbc-handler.json \
 --trustAll \
 --no-prompt
```

5. (Optional) For HTTP access logging, create an External HTTP Access Log Publisher

   The following example creates a JDBC HTTP access log publisher:

   ```
   $ dsconfig \
    create-log-publisher \
    --port 4444 \
    --hostname opendj.example.com \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --publisher-name "JDBC HTTP Access Log Publisher" \
    --type external-http-access \
    --set enabled:true \
    --set config-file:config/jdbc-handler.json \
    --trustAll \
    --no-prompt
   ```

6. (Optional) For a custom access logger, follow these general steps:

   a. Copy the .jar file for the custom audit event handler to `/path/to/opendj/lib/extensions`.

   b. Prepare the JSON configuration file for the custom handler.

   c. Create an External Access Log Publisher or External HTTP Access Log Publisher configuration as appropriate for the custom access logger.

To Trust Transaction IDs

Client applications using the Open Identity Platform 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, OpenDJ directory 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`.

- Set the advanced global server property, `trust-transaction-ids`, to `true`:

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

At this point transaction IDs are trusted, and can be written to the logs.

Elasticsearch Audit Event Handler Configuration

An Elasticsearch audit event handler logs audit event messages to an Elasticsearch server. This section briefly describes the JSON configuration file for the handler.

The JSON file 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".
    "connection": {
      "host": string,               // Elasticsearch host. Default: localhost
      "port": number,               // Elasticsearch host. Default: 9200
      "useSSL": boolean,            // Connect to Elasticsearch over HTTPS?
      "username": string,           // (Optional) User name for HTTP Basic auth.
      "password": string            // (Optional) Password for HTTP Basic auth.
    },
    "indexMapping": {
      "indexName": string           // Name of the Elasticsearch index.
    },
    "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.
    }
  }
}
```

Using an Elasticsearch Audit Log Handler

This example demonstrates logging an HTTP audit event message 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 OpenDJ HTTP audit event messages:

```
$ curl --request POST --header "Content-Type: application/json" --data '{
  "settings": {},
  "mappings": {
    "ldap-access": {
      "_source": {
        "enabled": true
      },
      "properties": {
        "timestamp": {
          "type": "date"
        },
        "eventName": {
          "type": "string",
          "index": "not_analyzed"
        },
        "transactionId": {
          "type": "string",
          "index": "not_analyzed"
        },
        "userId": {
          "type": "string",
          "index": "not_analyzed"
        },
        "trackingIds": {
          "type": "string",
          "index": "not_analyzed"
        },
        "server": {
          "properties": {
            "ip": {
              "type": "string",
              "index": "not_analyzed"
            },
            "port": {
              "type": "integer"
            }
          }
        },
        "client": {
          "properties": {
            "ip": {
              "type": "string",
              "index": "not_analyzed"
            },
            "port": {
```

```
            "type": "integer"
          }
        }
      },
      "request": {
        "properties": {
          "protocol": {
            "type": "string",
            "index": "not_analyzed"
          },
          "operation": {
            "type": "string",
            "index": "not_analyzed"
          },
          "detail": {
            "type": "nested"
          }
        }
      },
      "ldap": {
        "properties": {
          "connId": {
            "type": "integer",
            "index": "not_analyzed"
          },
          "msgId": {
            "type": "integer"
          },
          "dn": {
            "type": "string"
          },
          "scope": {
            "type": "string"
          },
          "filter": {
            "type": "string"
          },
          "attrs": {
            "type": "string"
          },
          "nentries": {
            "type": "string"
          },
          "authType": {
            "type": "string"
          },
          "reqControls": {
            "type": "string"
          },
          "respControls": {
            "type": "string"
          },
          "additionalItems": {
            "type": "string"
          },
          "items": {
            "type": "string"
          },
          "attr": {
            "type": "string"
          },
          "failureReason": {
            "type": "string"
          },
          "idToAbandon": {
            "type": "integer"
          },
          "maskedResult": {
            "type": "integer"
          },
          "maskedMessage": {
            "type": "string"
          },
          "message": {
            "type": "string"
          },
          "name": {
            "type": "string"
          },
          "newRDN": {
            "type": "string"
          },
          "newSup": {
            "type": "string"
          },
          "deleteOldRDN": {
            "type": "boolean"
          },
          "oid": {
            "type": "string"
          },
          "version": {
            "type": "string"
          },
          "reason": {
            "type": "string"
          },
          "opType": {
            "type": "string"
          }
        }
      },
      "response": {
        "properties": {
          "status": {
            "type": "string",
            "index": "not_analyzed"
          },
          "statusCode": {
            "type": "string",
            "index": "not_analyzed"
          },
          "detail": {
            "type": "string",
            "index": "not_analyzed"
          },
          "elapsedTime": {
            "type": "integer"
          },
          "elapsedTimeUnits": {
            "type": "string",
            "index": "not_analyzed"
          }
        }
      }
    }
  },
  "http-access": {
    "_source": {
      "enabled": true
    },
    "properties": {
      "timestamp": {
        "type": "date"
      },
      "eventName": {
        "type": "string",
        "index": "not_analyzed"
      },
      "transactionId": {
        "type": "string",
        "index": "not_analyzed"
      },
      "userId": {
        "type": "string",
        "index": "not_analyzed"
      },
      "trackingIds": {
        "type": "string",
        "index": "not_analyzed"
      },
      "server": {
        "properties": {
```

```
            "ip": {
              "type": "string",
              "index": "not_analyzed"
            },
            "port": {
              "type": "integer"
            }
          }
        },
        "client": {
          "properties": {
            "ip": {
              "type": "string",
              "index": "not_analyzed"
            },
            "port": {
              "type": "integer"
            }
          }
        },
        "request": {
          "properties": {
            "protocol": {
              "type": "string",
              "index": "not_analyzed"
            },
            "operation": {
              "type": "string",
              "index": "not_analyzed"
            },
            "detail": {
              "type": "nested"
            }
          }
        },
        "http": {
          "properties": {
            "request": {
              "properties": {
                "secure": {
                  "type": "boolean"
                },
                "method": {
                  "type": "string",
                  "index": "not_analyzed"
                },
                "path": {
                  "type": "string",
                  "index": "not_analyzed"
                },
                "queryParameters": {
```

```
                "type": "nested"
              },
              "headers": {
                "type": "nested"
              },
              "cookies": {
                "type": "nested"
              }
            }
          },
          "response": {
            "properties": {
              "headers": {
                "type": "nested"
              }
            }
          }
        }
      },
      "response": {
        "properties": {
          "status": {
            "type": "string",
            "index": "not_analyzed"
          },
          "statusCode": {
            "type": "string",
            "index": "not_analyzed"
          },
          "detail": {
            "type": "string",
            "index": "not_analyzed"
          },
          "elapsedTime": {
            "type": "integer"
          },
          "elapsedTimeUnits": {
            "type": "string",
            "index": "not_analyzed"
          }
        }
      }
    }
  }
}' http://localhost:9200/audit
{"acknowledged":true}
```

3. Configure OpenDJ directory server 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/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 OpenDJ directory server to use the Elasticsearch audit handler:

```
$ dsconfig \
 create-log-publisher \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "Elasticsearch HTTP Access Log Publisher" \
 --type external-http-access \
 --set enabled:true \
 --set config-file:config/elasticsearch-handler.json \
 --trustAll \
 --no-prompt
```

With Elasticsearch and OpenDJ diretory server running, audit event messages for HTTP requests to OpenDJ directory 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
{
  "_id": "bjensen",
  "_rev": "00000000828dc352",
  "schemas": ["urn:scim:schemas:core:1.0"],
  "userName": "bjensen@example.com",
  "displayName": "Barbara Jensen",
  "name": {
    "givenName": "Barbara",
    "familyName": "Jensen"
  },
  "contactInformation": {
    "telephoneNumber": "+1 408 555 1862",
    "emailAddress": "bjensen@example.com"
  },
  "meta": {},
  "manager": [{
    "_id": "trigden",
    "displayName": "Torrey Rigden"
  }]
}
```

A search request to Elasticsearch shows the resulting audit event content:

```
$ curl 'localhost:9200/audit/_search?q=*&pretty'
{
  "took" : 31,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "audit",
      "_type" : "http-access",
      "_id" : "a5c09e11-cc79-4a34-8dbe-b23cc1a79a8b-30",
      "_score" : 1.0,
      "_source" : {
        "eventName" : "OpenDJ Server-HTTP-ACCESS",
        "timestamp" : "2016-06-07T21:19:23.939Z",
        "transactionId" : "a5c09e11-cc79-4a34-8dbe-b23cc1a79a8b-29",
        "server" : {
          "ip" : "0:0:0:0:0:0:0:1",
          "port" : 8080
        },
        "client" : {
          "ip" : "0:0:0:0:0:0:0:1",
          "port" : 58907
        },
        "http" : {
          "request" : {
            "secure" : false,
            "method" : "GET",
            "path" : "http://opendj.example.com:8080/api/users/bjensen",
            "queryParameters" : { },
            "cookies" : { }
          },
          "response" : {
            "headers" : {
              "Cache-Control" : [ "no-cache" ],
              "Content-Type" : [ "application/json; charset=UTF-8" ],
              "ETag" : [ "\"00000000828dc352\"" ]
            }
          }
        },
        "response" : {
          "status" : "SUCCESSFUL",
          "statusCode" : "200",
          "elapsedTime" : 6,
          "elapsedTimeUnits" : "MILLISECONDS"
        }
      }
    } ]
  }
}
```

See the Elasticsearch documentation for details on searching and search results.

JDBC Audit Event Handler Configuration

The JDBC audit event handler that responds to events by logging messages to an appropriately configured relational database table. This section briefly describes the JSON configuration file for the handler.

The JSON file has the following format:

```
{
    "class": "org.forgerock.audit.handlers.jdbc.JdbcAuditEventHandler",
    "config": {
        "name": string,
        "topics": array,
        "databaseType": string,
        "enabled": boolean,
        "buffering": {
            "enabled": boolean,
            "writeInterval": duration,
            "autoFlush": boolean,
            "maxBatchedEvents": number,
            "maxSize": number,
            "writerThreads": number
        },
        "connectionPool": {
            "dataSourceClassName": string,
            "jdbcUrl": string,
            "username": string,
            "password": string,
            "autoCommit": boolean,
            "connectionTimeout": number,
            "idleTimeout": number,
            "maxLifetime": number,
            "minIdle": number,
            "maxPoolSize": number,
            "poolName": string
        },
        "tableMappings": [
            {
                "event": string,
                "table": string,
                "fieldToColumn": {
                    "event-field": "database-column"
                }
            }
        ]
    }
}
```

The `class` field identifies the handler.

The `"config"` object has the following properties:

`"name"`: ***string, required***

The name of the event handler.

`"topics"`: ***array of strings, required***

The topics that this event handler intercepts.

OpenDJ supports handling access events that occur at the system boundary, such as arrival of the initial request and departure of the final response.

Set this to `"topics": [ "http-access" ]` or `"topics": [ "ldap-access" ]`.

`"databaseType"`: ***string, required***

The database type name.

Built-in support is provided for `oracle`, `mysql`, and `h2`. Unrecognized database types rely on a `GenericDatabaseStatementProvider`.

`"enabled"`: ***boolean, optional***

Whether this event handler is active.

Default: true.

`"buffering"`: ***object, optional***

Buffering settings for sending messages to the database. The default is for messages to be written to the log file for each event.

The buffering object has the following fields:

`"enabled"`: ***boolean, optional***

Whether log buffering is enabled.

Default: false.

`"writeInterval"`: ***duration, required***

The interval at which to send buffered event messages to the database.

This interval must be greater than 0 if buffering is enabled.

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

`"autoFlush"`: ***boolean, optional***

Whether the events are automatically flushed after being written.

Default: true.

`"maxBatchedEvents"`: ***number, optional***

The maximum number of event messages batched into a PreparedStatement.

Default: 100.

`"maxSize"`: ***number, optional***

The maximum size of the queue of buffered event messages.

Default: 5000.

`"writerThreads"`: ***number, optional***

The number of threads to write buffered event messages to the database.

Default: 1.

`"connectionPool"`: ***object, required***

Connection pool settings for sending messages to the database.

The connection pool object has the following fields:

`"dataSourceClassName"`: ***string, optional***

The class name of the data source for the database.

`"jdbcUrl"`: ***string, required***

The JDBC URL to connect to the database.

`"username"`: ***string, required***

The username identifier for the database user with access to write the messages.

`"password"`: ***number, optional***

The password for the database user with access to write the messages.

`"autoCommit"`: ***boolean, optional***

Whether to commit transactions automatically when writing messages.

Default: true.

`"connectionTimeout"`: ***number, optional***

The number of milliseconds to wait for a connection from the pool before timing out.

Default: 30000.

`"idleTimeout"`: ***number, optional***

The number of milliseconds to allow a database connection to remain idle before timing out.

Default: 600000.

`"maxLifetime"`: ***number, optional***

The number of milliseconds to allow a database connection to remain in the pool.

Default: 1800000.

`"minIdle"`: ***number, optional***

The minimum number of idle connections in the pool.

Default: 10.

`"maxPoolSize"`: ***number, optional***

The maximum number of connections in the pool.

Default: 10.

`"poolName"`: ***string, optional***

The name of the connection pool.

`"tableMappings"`: ***array of objects, required***

Table mappings for directing event content to database table columns.

A table mappings object has the following fields:

`"event"`: ***string, required***

The audit event that the table mapping is for.

Set this to `access`.

`"table"`: ***string, required***

The name of the database table that corresponds to the mapping.

`"fieldToColumn"`: ***object, required***

This object maps the names of audit event fields to database columns, where the keys and values are both strings.

Audit event fields use JSON pointer notation, and are taken from the JSON schema for the audit event content.

Syslog Audit Event Handler Configuration

The Syslog audit event handler that responds to events by logging messages to the UNIX system log as governed by RFC 5424, The Syslog Protocol. This section briefly describes the JSON configuration file for the handler.

The JSON file has the following format:

```
{
    "class": "org.forgerock.audit.handlers.syslog.SyslogAuditEventHandler",
    "config": {
        "name": string,
        "topics": array,
        "protocol": string,
        "host": string,
        "port": number,
        "connectTimeout": number,
        "facility": "string",
        "buffering": {
            "enabled": boolean,
            "maxSize": number
        },
        "severityFieldMappings": [
            {
                "topic": string,
                "field": string,
                "valueMappings": {
                    "field-value": "syslog-severity"
                }
            }
        ]
    }
}
```

The `class` field identifies the handler.

The `"config"` object has the following properties:

`"name"`: ***string, required***

The name of the event handler.

`"topics"`: ***array of strings, required***

The topics that this event handler intercepts.

OpenDJ supports handling access events that occur at the system boundary, such as arrival of the initial request and departure of the final response.

Set this to `"topics": [ "http-access" ]` or `"topics": [ "ldap-access" ]`.

`"protocol"`: ***string, required***

The transport protocol used to send event messages to the Syslog daemon.

Set this to `TCP` for Transmission Control Protocol, or to `UDP` for User Datagram Protocol.

`"host"`: ***string, required***

The hostname of the Syslog daemon to which to send event messages. The hostname must resolve to an IP address.

`"port"`: ***number, required***

The port of the Syslog daemon to which to send event messages.

The value must be between 0 and 65535.

`"connectTimeout"`: ***number, required when using TCP***

The number of milliseconds to wait for a connection before timing out.

`"facility"`: ***string, required***

The Syslog facility to use for event messages.

Set this to one of the following values:

`kern`
: Kernel messages

`user`
: User-level messages

`mail`
: Mail system

`daemon`
: System daemons

`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

`"buffering"`: ***object, optional***

Buffering settings for writing to the system log facility. The default is for messages to be written to the log for each event.

The buffering object has the following fields:

`"enabled"`: ***boolean, optional***

Whether log buffering is enabled.

Default: false.

`"maxSize"`: ***number, optional***

The maximum number of buffered event messages.

Default: 5000.

`"severityFieldMappings"`: ***object, optional***

Severity field mappings set the correspondence between audit event fields and Syslog severity values.

The severity field mappings object has the following fields:

`"topic"`: ***string, required***

The audit event topic to which the mapping applies.

Set this to `access`.

`"field"`: ***string, required***

The audit event field to which the mapping applies.

Audit event fields use JSON pointer notation, and are taken from the JSON schema for the audit event content.

`"valueMappings"`: ***object, required***

The map of audit event values to Syslog severities, where both the keys and the values are strings.

Syslog severities are one of the following values:

`emergency`

System is unusable.

`alert`

Action must be taken immediately.

`critical`

Critical conditions.

`error`

Error conditions.

`warning`

Warning conditions.

`notice`

Normal but significant condition.

`informational`

Informational messages.

`debug`

Debug-level messages.

Error Logs

The *errors log* traces server events, error conditions, and warnings, categorized and identified by severity.

The following `errors` log excerpt shows log entries for a backup task, with lines wrapped for readability:

```
[06/Oct/2015:16:58:15 +0200] category=... severity=NOTICE msgID=...
 msg=Backup task 20151006165815904 started execution
[06/Oct/2015:16:58:15 +0200] category=TASK severity=NOTICE msgID=...
```

```
 msg=Starting backup for backend userRoot
[06/Oct/2015:16:58:16 +0200] category=UTIL severity=NOTICE msgID=...
 msg=Archived backup file: dj
...
[06/Oct/2015:16:58:16 +0200] category=UTIL severity=NOTICE msgID=...
 msg=Archived backup file: tasks.ldif
[06/Oct/2015:16:58:16 +0200] category=TASK severity=NOTICE msgID=...
 msg=The backup process completed successfully
[06/Oct/2015:16:58:16 +0200] category=... severity=NOTICE msgID=...
 msg=Backup task 20151006165815904 finished execution in the state
     Completed successfully
```

HTTP Access Logs

For the HTTP Connection Handler, OpenDJ maintains a separate access log in `logs/http-access`. This access log, by default configured as the File Based HTTP Access Log Publisher, uses a different format than the LDAP access log. This HTTP access log uses Extended Log File Format with fields described in Microsoft's implementation as well.

Interface stability: Evolving (See "Product Interface Stability" in the *Reference*)

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

  This field is new in 3.5.

`cs-uri-stem`
: URL-encoded path requested by the client

  This field is new in 3.5.

`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`

Open Identity Platform common audit event framework transaction ID for the request

This defaults to `0` unless you configure OpenDJ to trust transaction IDs as described in "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 the default configuration. Lines are folded and space reformatted for the printed page:

```
-  192.168.0.15  bjensen   22/May/2013:10:06:18 +0200
  GET  /users/bjensen?_prettyPrint=true                      HTTP/1.1    200
  curl/7.21.4  3    40
-  192.168.0.15  bjensen   22/May/2013:10:06:52 +0200
  GET  /groups/Directory%20Administrators?_prettyPrint=true  HTTP/1.1    200
  curl/7.21.4  4    41
-  192.168.0.12  bjensen   22/May/2013:10:07:07 +0200
  GET  /users/missing?_prettyPrint=true                      HTTP/1.1    200
  curl/7.21.4  5     9
-  192.168.0.12  -         22/May/2013:10:07:46 +0200
  GET  /users/missing?_prettyPrint=true                      HTTP/1.1    401
  curl/7.21.4  6     0
-  192.168.0.15  kvaughan  22/May/2013:10:09:10 +0200
  POST /users?_action=create&_prettyPrint=true           HTTP/1.1    200
  curl/7.21.4  7   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

Replication Logs

The *replication log* traces replication events, with entries similar to the errors log. The following excerpt has lines wrapped for readability:

```
[22/Jun/2011:14:37:34 +0200] category=SYNC severity=NOTICE msgID=15139026
msg=Finished total update: exported domain "dc=example,dc=com" from this
directory server DS(24065) to all remote directory servers.
[22/Jun/2011:14:37:35 +0200] category=SYNC severity=MILD_WARNING msgID=14745663
msg=Replication server RS(23947) at opendj.example.com/10.10.0.168:8989 has
closed the connection to this directory server DS(24065). This directory
server will now try to connect to another replication server in order to
receive changes for the domain "dc=example,dc=com"
[22/Jun/2011:14:37:35 +0200] category=SYNC severity=NOTICE msgID=15138894
msg=The generation ID for domain "dc=example,dc=com" has been reset to 3679640
```

Notice that the replication log does not trace replication operations. Use the external change log instead to get notifications about changes to directory data over protocol. You can alternatively configure an audit log, which is a type of access log that dumps changes in LDIF.

Debug Logs

A *debug log* traces details needed to troubleshoot a problem in the server. 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.

Log Rotation and Retention

Each file-based log can be associated with a *log rotation policy*, and a *log retention policy*. The former can specify when, after how much time, or at what maximum size a log is rotated. The latter can specify a maximum number or size of logs to retain, or an amount of free disk space to maintain. The design allows for custom policies as well.

By default the file-based logs are subject to rotation and retention policies that you can list with `dsconfig list-log-rotation-policies` and `dsconfig list-log-retention-policies`.

For example, view the log rotation policies with the following command:

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


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          : -
: -
```

View the log retention policies with the following command:

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


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          : -
```

```
: -
```

Use the `dsconfig get-log-publisher-prop` command to examine the policies that apply to a particular logger:

```
$ dsconfig \
 get-log-publisher-prop \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Access Logger" \
 --property retention-policy \
 --property rotation-policy
Property         : Value(s)
-----------------:-------------------------------------------------------------
retention-policy : File Count Retention Policy
rotation-policy  : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                 : Policy
```

In other words, by default OpenDJ keeps 10 access log files, rotating the access log each day, or when the log size reaches 100 MB.

The `dsconfig` command offers a number of subcommands for creating and deleting log rotation and retention policies, and for setting policy properties. You can update which policies apply to a logger by using the `dsconfig set-log-publisher-prop` command.

Log Filtering

Each time a client application sends a request to OpenDJ, the server writes to its access log. As shown above, a simple search operation results in five messages written to the 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 limit what the server logs, and focus on what you want to see. You define the filter criteria, and also set the filtering policy.

You can filter both access and also audit logs. Log filtering lets you define rules based these criteria:

- Client IP address, bind DN, group membership
- Port number
- Protocol used (such as LDAP, LDAPS, JMX)
- Response times
- Result codes (only log error results, for example)
- Search response criteria (number of entries returned, whether the search was indexed)
- Target DN

- Type of operation (connect, bind, add, delete, modify, rename, search, etc.)

The filtering policy in the log publisher configuration specifies whether to include or exclude log messages that match the criteria you define. OpenDJ does not filter logs until you update the log publisher configuration.

Example: Exclude Control Panel-Related Messages

A common development troubleshooting technique consists of sending client requests while tailing the access log:

```
$ tail -f /path/to/opendj/logs/access
```

The trouble is, when OpenDJ control panel is running, or when you are also adapting your configuration using the `dsconfig` command, OpenDJ writes access log messages related to administration. These might prevent you from noticing the messages that interest you.

This example demonstrates how to filter out access log messages due to administrative connections over LDAPS on ports 1636 and 4444.

Create access log filtering criteria rules:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Access Logger" \
 --criteria-name "Exclude LDAPS on 1636 and 4444" \
 --type generic \
 --set connection-port-equal-to:1636 \
 --set connection-port-equal-to:4444 \
 --set connection-protocol-equal-to:ldaps \
 --trustAll \
 --no-prompt
```

Activate filtering to exclude messages from the default access log according to the criteria you specified:

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

At this point, OpenDJ filters out connections over LDAPS to ports 1636 and 4444. While performing operations in OpenDJ control panel, if you perform a simple `ldapsearch --port 1389 --baseDN dc=example,dc=com uid=bjensen cn`, then all you see in the access log is the effect of the `ldapsearch` command:

```
$ tail -f /path/to/opendj/logs/access
[19/Oct/2011:16:37:16 +0200] CONNECT conn=8 from=127.0.0.1:54165
 to=127.0.0.1:1389 protocol=LDAP
[19/Oct/2011:16:37:16 +0200] SEARCH REQ conn=8 op=0 msgID=1
 base="dc=example,dc=com" scope=wholeSubtree filter="(uid=bjensen)" attrs="cn"
[19/Oct/2011:16:37:16 +0200] SEARCH RES conn=8 op=0 msgID=1 result=0 nentries=1
 etime=14
[19/Oct/2011:16:37:16 +0200] UNBIND REQ conn=8 op=1 msgID=2
[19/Oct/2011:16:37:16 +0200] DISCONNECT conn=8 reason="Client Unbind"
```

In addition to the filtering policy, you can also adjust how OpenDJ writes log messages. By default, OpenDJ writes one log message for a request, and another for a response. You can set the log publisher property `log-format` to `combined` to have OpenDJ write a single message per operation. This can be helpful, for example, when evaluating response times. In addition, you can change the log message time stamps with `log-record-time-format`, and specify whether to log LDAP control OIDs for operations by setting `log-control-oids` to `true`.

Alert Notifications

OpenDJ can send alerts to provide notifications of significant server events. Yet alert notifications are not enabled by default. You can use the `dsconfig` command to enable alert notifications:

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

OpenDJ can also send mail over SMTP instead of JMX notifications. Before you set up the SMTP-based alert handler, you must identify an SMTP server to which OpenDJ sends messages:

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

$ dsconfig \
 create-alert-handler \
 --port 4444 \
 --hostname opendj.example.com \
 --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 directory server uses the following types when sending alerts. For alert types that indicate server problems, check `OpenDJ/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 directory 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 directory 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 directory server is unable to rename the new tasks backing file into place.

`org.opends.server.CannotScheduleRecurringIteration`

The directory server is unable to schedule an iteration of a recurring task.

`org.opends.server.CannotWriteConfig`

The directory 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 directory server is unable to write an updated tasks backing file for some reason.

`org.opends.server.DirectoryServerShutdown`

The directory server has begun the process of shutting down.

`org.opends.server.DirectoryServerStarted`

The directory 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 directory 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 that 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 directory server is leaving lockdown mode.

`org.opends.server.ManualConfigEditHandled`

The directory 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 directory 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 directory 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.

[1] You can also configure the access logger to combine log messages 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.

Tuning Servers For Performance

This chapter suggests ways to measure and improve directory service performance. In this chapter you will learn to:

- Define directory server 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 OpenDJ.

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.

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.

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 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.

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.

Server Hardware Recommendations

OpenDJ runs on systems with Java support, and is therefore very portable. OpenDJ tends to perform best on single-board, x86 systems due to low memory latency.

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 server is 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 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.

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 OpenDJ as you will later in production, and then conduct experiments to determine how best to meet the requirements defined in the SLA.

Use the `make-ldif` command, described in makeldif(1) in the *Reference*, to generate sample data that match what you expect to find in production.

The OpenDJ LDAP Toolkit provides command-line tools to help with basic performance testing:

- The `addrate` command measures add and delete throughput and response time.
- The `authrate` command measures bind throughput and response time.
- The `modrate` command measures modification throughput and response time.
- The `searchrate` command 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set etime-resolution:nanoseconds \
 --trustAll \
 --no-prompt
```

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.

Maximum Open Files

OpenDJ 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 OpenDJ.

When setting up OpenDJ for production use, make sure OpenDJ can use at least 64K (65536) file descriptors. For example, when running OpenDJ 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
```

Java Settings

Default Java settings let you evaluate OpenDJ 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`, and apply the changes with the `dsjavaproperties` command, described in dsjavaproperties(1) in the *Reference*:

`-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`

When using CMS garbage collection, consider using this option. Do not use it when using G1 garbage collection.

If a 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 (`-Xmn512M`).

`-XX:MaxTenuringThreshold=1`

Force OpenDJ directory server to only create objects that have either a short lifetime, or a long lifetime.

`-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.

Data Storage Settings

By default, OpenDJ compresses attribute descriptions and object class sets to reduce data size. This is called compact encoding.

By default, OpenDJ does 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` OpenDJ compresses entries before writing them to the database:[1]

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

$ import-ldif \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --ldifFile /path/to/Example.ldif \
 --backendID userRoot \
 --includeBranch dc=example,dc=com \
 --start 0
Import task 20120917100628767 scheduled to start Sep 17, 2012 10:06:28 AM CEST
```

If write traffic to your directory service occurs in short bursts, and you use database backends of type `pdb`, you can potentially improve short-term performance during the bursts by increasing the `db-checkpointer-wakeup-interval` setting. This setting specifies the maximum length of time between attempts to write a checkpoint to the journal. Longer intervals allow more updates to accumulate in buffers before they are required to be written to disk. The transaction log is still written to disk, but the modified pages are kept in memory longer before being written. Longer intervals potentially cause recovery from an abrupt termination to take more time.

LDIF Import Settings

You can tweak OpenDJ 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 OpenDJ. 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 from OpenDJ with all checks active, for example, you can skip schema validation. Use the `--skipSchemaValidation` option with the `import-ldif` command to skip validation.

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. Database backends of type pdb allocate all of the cache memory at startup.
- 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.

Caching Large, Frequently Used Entries

OpenDJ implements 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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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.

Logging Settings

Debug logs trace the internal workings of OpenDJ, and therefore generally should be used sparingly, especially 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, as each client request results in multiple access log messages. Consider disabling the access log in such cases:

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

[1] OpenDJ does not proactively rewrite all entries in the database after you change the settings. Instead, to force OpenDJ to compress all entries, import the data from LDIF.

Securing and Hardening OpenDJ Directory Server

By default OpenDJ directory server is set up for ease of evaluation and deployment. When you deploy OpenDJ 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 OpenDJ directory server, and appropriately protect access to directory 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.

Set Up a System Account for OpenDJ Directory Server

Do not run OpenDJ directory 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 OpenDJ directory server, you can use system controls to limit user access.

The user running OpenDJ directory 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 OpenDJ directory 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 lets users in the same group as the user running the server read server files, though not directory data files.

The user running OpenDJ directory server does not, however, need access to login from a remote system or to perform actions unrelated to OpenDJ.

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.

On UNIX and Linux, the group for the user running OpenDJ directory server has access by default to read files, including log files. You can restrict this after installation by setting the `log-file-permissions` property on each active log publisher.

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 OpenDJ directory server as a Windows service. For details see windows-service(1) in the *Reference*.

Install and Use Java Security Updates

Security updates are occasionally 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.

After you update the Java runtime environment, edit the `default.java-home` setting in the file `/path/to/opendj/config/java.properties` to use the path to the update release, and then use the `dsjavaproperties` command for the changes to be taken into account. Then restart OpenDJ directory server. For details see dsjavaproperties(1) in the *Reference*.

Only Enable Necessary Services

By default, OpenDJ directory 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.

Configure Logging Appropriately

By default, OpenDJ directory 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 directory 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 in OpenDJ directory server is not for security audits. Instead it records changes in LDIF. The audit log is intended to help you as server administrator 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".

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 directory servers and directory data, and you make it easier to audit what administrators can do.

Reconsider Default Global ACIs

Global ACIs are defined in the directory server configuration. Global ACIs apply whenever no other ACIs take precedence. Global ACIs allow applications to read the root DSE, to read directory server schema, to read directory data anonymously, to modify one's own entry, and to request extended operations and operations with certain controls. For details see "Default Global ACIs".

If the default global ACIs 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. For details about ACIs see "Configuring Privileges and Access Control".

Generally it is fine to allow applications at least to read the root DSE and schema operational attributes, 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 schema operational attributes describe the data stored in the directory. The StartTLS extended operation lets an application initiate a secure session starting on a port that does not require encryption.

Protect Directory Server Network Connections

Directory 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 be able to inspect the network traffic without decrypting messages. For final testing and production environments, secure the connections.

Transport layer security depends on public key infrastructure when negotiating encryption. OpenDJ directory server has multiple keystores and truststores for handling the key pairs and public key certificates as described in "Changing Server Certificates".

OpenDJ directory 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 ACI to require secure communications for most operations. Keep a global ACI that allows anonymous access to request the StartTLS extended operation. For all operations other than requesting StartTLS, use ACIs whose subject sets `authmethod = ssl`, and also sets `ssf` appropriately.

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 OpenDJ directory 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 OpenDJ directory server and a client application negotiate connection security, they must agree on a security protocol and cipher suite. By default OpenDJ directory 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.

Use Appropriate Password Storage and Password Policies

Make sure you keep passwords secret in production. OpenDJ directory 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.

Passwords in Configuration Files

OpenDJ directory 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 OpenDJ directory 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.

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 \
 --bindDN "cn=Directory Manager" \
 --bindPassword - \
 --port 1389 \
 --hostname opendj.example.com \
 --baseDN cn=config \
 "(cn=Directory Manager)" \
 userPassword
```

```
Password for user 'cn=Directory Manager':
dn: cn=Directory Manager,cn=Root DNs,cn=config
userPassword: {SSHA512}WiYWHyAa612EZwCMY7uGwN/WYp2Ne7EmV0QTPX5g6RrTKi8jZX3u5rBIW
 OUY1DPK3TGYqDiF7d/BEhHnIjBmBtkotWkHIKMa
```

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.

Passwords in Directory Data

OpenDJ directory 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 OpenDJ directory 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 provides a wide range of alternatives. For details see "Configuring Password Policy".

Protect OpenDJ Directory Server Files

By default, OpenDJ directory server does not encrypt directory 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 "Set Up a System Account for OpenDJ Directory Server", and unpacked the server files as that user, then the system should prevent other users from having overly permissive access to directory server files.

Included in the files that directory 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 \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword - \
 --backupAll \
 --backupDirectory /path/to/opendj/bak \
 --encrypt \
 --start 0
Password for user 'cn=Directory Manager':
Backup task 20150810105606755 scheduled to start ...
```

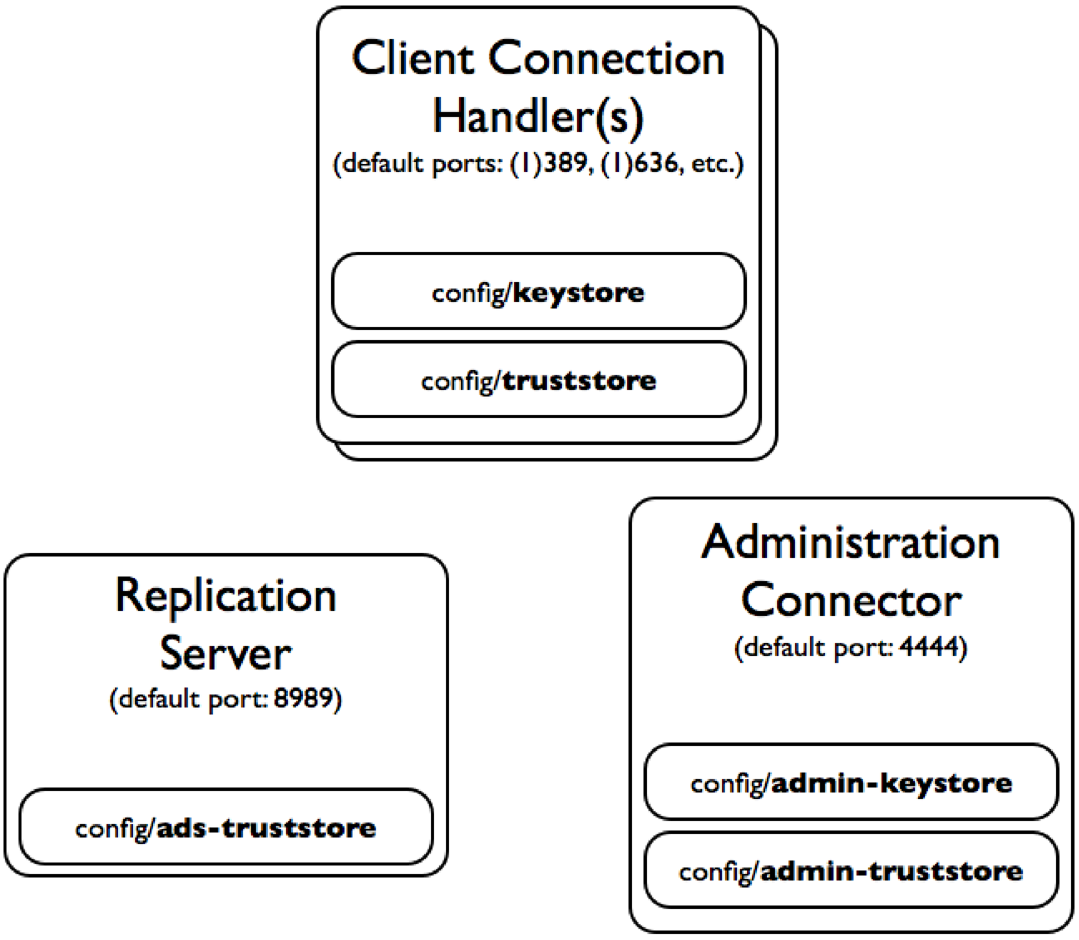
The server uses its Crypto Manager configuration to determine how to encrypt the backup archive data. The --encrypt option is not available for offline back up. If you back up server data offline, plan to protect the files separately.

Changing Server Certificates

This chapter covers how to replace OpenDJ key pairs and public key certificates. In this chapter you will learn to:

- Replace a key pair for securing a connection handler
- Replace a key pair used for replication

OpenDJ uses keystores (for private keys) and truststores (for public, signed certificates). Up to three sets of keystores are used, as shown in the following illustration.



By default the keystores are located in the `/path/to/opendj/config` directory:

- The `keystore` and `truststore` hold keys for securing connections with client applications.
- The `admin-keystore` and `admin-truststore` hold keys for securing administrative connections, such as those used when connecting with the `dsconfig` command.
- The `ads-truststore` holds keys for securing replication connections with other OpenDJ servers in the replication topology.

Each keystore has a specific purpose:

`admin-keystore`

This Java Keystore holds the private key and administrative certificate for the server, `admin-cert`.

This key pair is used to protect communications on the administration port. The password, stored in `admin-keystore.pin`, is also the key password for `admin-cert`.

`admin-truststore`

This Java Keystore holds a copy of the administrative certificate, `admin-cert`. The password is the same as for the `admin-keystore`, in other words the string in `admin-keystore.pin`.

`ads-truststore`

This Java Keystore holds public key certificates of all servers replicating with the current server. It also includes the `ads-certificate` key pair of the current server. The password is stored in `ads-truststore.pin`.

Do not change this keystore directly.

`keystore`

This Java Keystore holds the private key and server certificate, `server-cert`, used to protect TLS/SSL communications with client applications. The password, stored in `keystore.pin`, is also the key password for `server-cert`.

`truststore`

This Java Keystore holds a copy of the `server-cert` certificate from the `keystore`. This is also where you import certificates of client applications if you want OpenDJ to recognize them. The password is the same as for the `keystore`, in other words the string in `keystore.pin`.

TIP

Examples in this chapter use self-signed certificates, but you can also use certificates signed by a Certificate Authority (CA).

When importing a certificate (`keytool -import`) signed by a well-known CA, use the `-trustcacerts` option to trust the CA certificates delivered with the Java runtime environment.

To Replace a Server Key Pair

This procedure shows how to replace a server key pair in the `admin-keystore` and copy of the administrative certificate in `admin-truststore`.

The examples also apply when replacing a key pair in the `keystore` and copy of the server certificate in `truststore`. Just adapt the commands to use the correct keystore, truststore, and PIN file names.

This procedure does not apply for replication key pairs. Instead, see "To Replace the Key Pair Used for Replication".

1. Check the alias of the key pair and certificate copy to replace:

```
$ cd /path/to/opendj/config
$ keytool -list -keystore admin-keystore -storepass `cat admin-keystore.pin`

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

admin-cert, May 20, 2015, PrivateKeyEntry,
Certificate fingerprint (SHA1):
21:9F:F0:E8:A3:22:A3:62:1D:C7:04:BD:12:44:A6:FA:0C:3F:3A:35
$ keytool -list -keystore admin-truststore -storepass `cat admin-keystore.pin`

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

admin-cert, May 20, 2015, trustedCertEntry,
Certificate fingerprint (SHA1):
21:9F:F0:E8:A3:22:A3:62:1D:C7:04:BD:12:44:A6:FA:0C:3F:3A:35
```

This alias is also stored in the server configuration.

2. Remove the key pair and certificate copy to replace:

```
$ keytool \
 -delete \
 -alias admin-cert \
 -keystore admin-keystore \
 -storepass `cat admin-keystore.pin`

$ keytool \
 -delete \
 -alias admin-cert \
 -keystore admin-truststore \
 -storepass `cat admin-keystore.pin`
```

3. Generate a new key pair in the keystore:

```
$ keytool \
 -genkey \
 -alias admin-cert \
 -keyalg RSA \
 -validity 7300 \
 -keysize 2048 \
 -ext "san=dns:opendj.example.com" \
 -dname "CN=opendj.example.com, O=Administration Connector Self-Signed
Certificate" \
 -keystore admin-keystore \
 -storepass `cat admin-keystore.pin` \
 -keypass `cat admin-keystore.pin`
```

Notice that the -alias option takes the same alias as before. This is because the ssl-cert-nickname for the Administration Connector is configured as admin-cert. Also, the -dname option has a CN value corresponding to the fully qualified domain name of the host where OpenDJ directory server is running.

4. Get the new key pair's certificate signed, using one of the following alternatives:

   ◦ Self-sign the certificate:

     ```
     $ keytool \
      -selfcert \
      -alias admin-cert \
      -validity 7300 \
      -keystore admin-keystore \
      -storepass `cat admin-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 Request and Install a CA-Signed Certificate".

5. Export a copy of the certificate from the keystore:

   ```
   $ keytool \
    -export \
    -alias admin-cert \
    -keystore admin-keystore \
    -storepass `cat admin-keystore.pin` \
    -file admin-cert.crt
   Certificate stored in file <admin-cert.crt>
   ```

6. Import the copy of the certificate into the truststore:

   ```
   $ keytool \
    -import \
    -alias admin-cert \
    -keystore admin-truststore \
    -storepass `cat admin-keystore.pin` \
    -file admin-cert.crt
   Owner: CN=opendj.example.com, O=Administration Connector Self-Signed
   Certificate
   Issuer: CN=opendj.example.com, O=Administration Connector Self-Signed
   Certificate
   Serial number: 4cdd42a
   Valid from: Thu May 28 11:32:05 CEST 2015 until: Wed May 23 11:32:05 CEST 2035
   Certificate fingerprints:
     MD5:  40:38:24:5D:DD:BE:EC:D6:07:56:08:25:95:D9:61:FE
   ```

```
  SHA1: BC:3D:A9:26:CD:4E:71:04:44:16:1E:A5:79:DA:43:2A:65:E8:85:85
  SHA256: D3:41:EE:44:5A:54:74:11:5A:...:9F:8F:08:13:09:DD:71:52:7E:35:66:7E
  Signature algorithm name: SHA256withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.17 Criticality=false
SubjectAlternativeName [
  DNSName: opendj.example.com
]

#2: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 08 E3 D3 62 AA 68 E6 02   52 25 F8 22 C4 43 82 2D  ...b.h..R%.".C.-
0010: 20 C1 39 99                                          .9.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

7. Restart OpenDJ to make sure it reloads the keystores:

```
$ cd /path/to/opendj/bin
$ stop-ds --restart
```

8. If you have client applications trusting the self-signed certificate, have them import the new one (`admin-cert.crt` in this example).

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
 QUFADA6MRswGQYDVQQKExJPcGVuREogQ2VydGlmaWNhdGUxGzAZBgNVBAMTEm9wZW5hbS5leGF
tcGxl
 LmNvbTAeFw0xMzAyMDcxMDMwMzNaFw0zMzAyMDIxMDMwMzNaMDoxGzAZBgNVBAoTEk9wZW5ESi
BDZXJ
 0aWZpY2F0ZTEbMBkGA1UEAxMSb3BlbmFtLmV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA
4GNAD
```

```
 CBiQKBgQCfGLAiUOz4sC8CM9T5DPTk9V9ErNC8N59XwBt1aN7UjhQl4/JZZsetubtUrZBLS9cR
rnYdZ
 cpFgLQNEmXifS+PdZ0DJkaLNFmd8ZX0spX8++fb4SkkggkmNRmi1fccDQ/DHMlwl7kk884lXum
mrzcD
 GbZ7p4vnY7y7GmD1vZSP+wIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAJciUzUP8T8A9VV6dQB0SY
CNG1o
 7IvpE7jGVZh6KvM0m5sBNX3wPbTVJQNij3TDm8nx6yhi6DUkpiAZfz/OBL5k+WSw80TjpIZ2+k
lhP1s
 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 `cat /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
 QUFADA6MRswGQYDVQQKExJPcGVuREogQ2VydGlmaWNhdGUxGzAZBgNVBAMTEm9wZW5hbS5leGF
tcGxl
 LmNvbTAeFw0xMzAyMDcxMDMwMzNaFw0zMzAyMDIxMDMwMzNaMDoxGzAZBgNVBAoTEk9wZW5ESi
BDZXJ
 0aWZpY2F0ZTEbMBkGA1UEAxMSb3BlbmFtLmV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA
4GNAD
 CBiQKBgQCfGLAiUOz4sC8CM9T5DPTk9V9ErNC8N59XwBt1aN7UjhQl4/JZZsetubtUrZBLS9cR
rnYdZ
 cpFgLQNEmXifS+PdZ0DJkaLNFmd8ZX0spX8++fb4SkkggkmNRmi1fccDQ/DHMlwl7kk884lXum
mrzcD
 GbZ7p4vnY7y7GmD1vZSP+wIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAJciUzUP8T8A9VV6dQB0SY
CNG1o
 7IvpE7jGVZh6KvM0m5sBNX3wPbTVJQNij3TDm8nx6yhi6DUkpiAZfz/OBL5k+WSw80TjpIZ2+k
lhP1s
 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 \
 --filename /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 "Configuring Replication":

```
$ 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
```

Moving Servers

This chapter explains how to move OpenDJ directory servers. 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 directory server

When you change where OpenDJ is deployed, you must take host names, port numbers, and certificates into account. The changes can also affect your replication configuration.

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 disabling replication for the server before the move, and then enabling and initializing replication again after the move.

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 disabling 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 \
 --port 4444 \
 --hostname opendj2.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set writability-mode:internal-only \
 --trustAll \
 --no-prompt
```

3. Disable replication for the server:

```
$ dsreplication \
 disable \
 --disableAll \
 --port 4444 \
 --hostname opendj2.example.com \
 --adminUID admin \
 --adminPassword password \
 --trustAll \
 --no-prompt
Establishing connections ..... Done.
Disabling replication on base DN dc=example,dc=com of server
 opendj2.example.com:4444 ..... Done.
Disabling replication on base DN cn=admin data of server
 opendj2.example.com:4444 ..... Done.
Disabling replication on base DN cn=schema of server
 opendj2.example.com:4444 ..... Done.
Disabling replication port 8989 of server opendj2.example.com:4444 ..... Done.
Removing registration information ..... Done.
Removing truststore information ..... Done.

See
/var/.../opends-replication-3173475478874782719.log
for a detailed log of this operation.
```

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
Stopping Server...

... msg=The Directory Server is now stopped
```

5. (Optional) You might also choose to remove extra log files from the server `logs/` directory

before moving the server.

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 enable 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
... The Directory Server has started successfully
```

5. Enable and initialize replication:

```
$ dsreplication \
 enable \
 --adminUID admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --host1 opendj.example.com \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
 --host2 opendj2.example.com \
 --port2 4444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 8989 \
 --trustAll \
 --no-prompt

Establishing connections ..... Done.
Checking registration information ..... Done.
Configuring Replication port on server opendj.example.com:4444 ..... Done.
Updating remote references on server opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN dc=example,dc=com on server
 opendj2.example.com:4444 ..... Done.
```

```
Updating registration configuration on server
 opendj.example.com:4444 ..... Done.
Updating registration configuration on server
 opendj2.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj.example.com:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
 opendj2.example.com:4444 ..... Done.
Initializing registration information on server opendj.example.com:4444 with
 the contents of server opendj2.example.com:4444 ..... Done.
Initializing schema on server opendj2.example.com:4444 with the contents of
 server opendj.example.com:4444 ..... Done.

Replication has been successfully enabled.  Note that for replication to work
 you must initialize the contents of the base DN's that are being replicated
 (use dsreplication initialize to do so).

See /tmp/opends-replication-1476402020764482023.log for a detailed log of this
operation.

$ dsreplication \
 pre-external-initialization \
 --adminUID admin \
 --bindPassword password \
 --port 4444 \
 --baseDN dc=example,dc=com \
 --trustAll \
 --no-prompt

Preparing base DN dc=example,dc=com to be initialized externally ..... Done.

Now you can proceed to the initialization of the contents of the base DN's on
 all the replicated servers.  You can use the command import-ldif or the binary
 copy to do so.  You must use the same LDIF file or binary copy on each server.

When the initialization is completed you must use the subcommand
 'post-external-initialization' for replication to work with the new base DN's
 contents.

$ dsreplication \
 post-external-initialization \
 --adminUID admin \
 --bindPassword password \
 --port 4444 \
 --baseDN dc=example,dc=com \
 --trustAll \
 --no-prompt

Updating replication information on base DN dc=example,dc=com ..... Done.
```

```
Post initialization procedure completed successfully.
```

6. Accept updates from client applications:

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

7. Direct client applications to the server.

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 directory 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 directory server when solving problems
- Troubleshoot problems with the way client applications access the directory
- Prepare evidence when asking a directory expert for help

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.

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 /var/....log for a detailed log of this operation.
```

Prevent antivirus and intrusion detection systems from interfering with OpenDJ directory server.

Antivirus and intrusion detection systems that do a deep inspection of database files are not compatible with OpenDJ directory server. Disable antivirus and intrusion detection systems, or at least prevent them from operating on OpenDJ directory server files.

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 directory server stores the entry for Directory Manager in the LDIF representation of its configuration. You must be able to edit directory 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
Encoded Password:  "{SSHA512}yWqHnYV4a5llPvE7WHLe5jzK27oZQWLIlVcs9gySu4TyZJMg
 NQNRtnR/Xx2xces1wu1dVLI9jVVtl1W4BVsmOKjyjr0rWrHt"
```

2. Stop OpenDJ directory server while you edit the configuration:

```
$ stop-ds
```

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: {SSHA512}yWqHnYV4a5llPvE7WHLe5jzK27oZQWLIlVcs9gySu4TyZJMg
 NQNRtnR/Xx2xces1wu1dVLI9jVVtl1W4BVsmOKjyjr0rWrHt
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 OpenDJ directory server again:

```
$ start-ds
```

5. Verify that you can administer the server as Directory Manager using the new password:

```
$ dsconfig -p 4444 -h opendj.example.com -D "cn=Directory Manager" -w password


>>>> OpenDJ configuration console main menu

What do you want to configure?

...

Enter choice: q
```

To Reset the Global Administrator's Password

When you enable 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 \
 --useStartTLS \
 --port 1389 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --authzID "cn=admin,cn=Administrators,cn=admin data" \
 --newPassword password
The LDAP password modify operation was successful
```

2. Let replication copy the password change to other replicas.

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 \
 --no-prompt \
 --trustAll
```

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" \
 --no-prompt \
 --trustAll

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 \
 --no-prompt \
 --trustAll
```

3. Restart OpenDJ to see debug messages in the log:

```
$ stop-ds --restart
...
$ dsconfig \
 list-debug-targets \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --no-prompt \
 --trustAll

Debug Target          : enabled : debug-exceptions-only
----------------------:---------:---------------------
org.opends.server.api : true    : false

$ tail -f /path/to/opendj/logs/debug
...
```

CAUTION OpenDJ directory server can generate a high volume of debug output. Use debug logging very sparingly on production systems.

Preventing Access While You Fix Issues

Misconfiguration can potentially put OpenDJ 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 provides a *lockdown mode* that allows connections only on the loopback address, and allows only operations requested by root users, such as `cn=Directory Manager`. You can use lockdown mode to prevent all but administrative access to OpenDJ in order to repair the server.

To put OpenDJ 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 OpenDJ on the local host):

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd
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
```

```
Processing ADD request for
 ds-task-id=Enter Lockdown Mode,cn=Scheduled Tasks,cn=tasks
ADD operation successful for DN
 ds-task-id=Enter Lockdown Mode,cn=Scheduled Tasks,cn=tasks
```

OpenDJ logs a notice message in `logs/errors` when lockdown mode takes effect:

```
[30/Jan/2012:17:04:32 +0100] category=BACKEND severity=NOTICE msgID=9896350
 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=*)" +
SEARCH operation failed
Result Code:  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:

```
$ ldapmodify \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --defaultAdd
dn: ds-task-id=Leave Lockdown Mode,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
ds-task-id: Leave Lockdown Mode
ds-task-class-name: org.opends.server.tasks.LeaveLockdownModeTask

Processing ADD request for
 ds-task-id=Leave Lockdown Mode,cn=Scheduled Tasks,cn=tasks
ADD operation successful for DN
 ds-task-id=Leave Lockdown Mode,cn=Scheduled Tasks,cn=tasks
```

OpenDJ also logs a notice message when leaving lockdown:

```
[30/Jan/2012:17:13:05 +0100] category=BACKEND severity=NOTICE msgID=9896350
 msg=Leave Lockdown task Leave Lockdown Mode finished execution
```

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 \
 --port 4444 \
 --hostname opendj.example.com \
 --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.

Troubleshooting TLS/SSL Connections

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 \
 | wc -l
 208
```

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
```

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:

```
[21/Nov/2011:13:03:20 -0600] category=SYNC severity=NOTICE
 msgID=15138921 msg=SSL connection attempt from myserver (123.456.789.012)
 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/admin-truststore \
 -storepass `cat config/admin-keystore.pin` |grep "^Owner:"

# Examine the certificates used for replication.
$ keytool -list -v -keystore config/ads-truststore \
 -storepass `cat 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/admin-truststore
-storepass `cat config/admin-keystore.pin` | grep "^Owner:"
Owner: CN=opendj.example.com, O=Administration Connector Self-Signed Certificate

$ keytool -list -v -keystore config/ads-truststore \
 -storepass `cat 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. Disable replication in each replica:

   ```
   $ dsreplication \
    disable \
    --disableAll \
    --port adminPort \
    --hostname hostName \
    --adminUID admin \
   ```

```
 --adminPassword password \
 --trustAll \
 --no-prompt
```

2. Stop and restart each server in order to clear the in-memory ADS truststore backend.
3. Enable replication making certain that fully qualified host names are used throughout:

```
$ dsreplication \
 enable \
 --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/admin-truststore \
 -storepass `cat config/admin-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".

Handling Compromised Keys

As explained in "Changing Server Certificates", OpenDJ directory 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, 2013:

     ```
     $ 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: 201303210834Z

Processing MODIFY request for ds-cfg-key-id=4F2F97979A7C05162CF64C9F73AF66ED,
 cn=instance keys,cn=admin data
MODIFY operation successful for DN ds-cfg-key-
id=4F2F97979A7C05162CF64C9F73AF66ED
 ,cn=instance keys,cn=admin data
```

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.

Troubleshooting Client Operations

By default OpenDJ logs information about all LDAP client operations in `logs/access`, and all HTTP client operations in `logs/http-access`. The following lines are wrapped for readability, showing a search for the entry with `uid=bjensen` as traced in the LDAP access log. In the access log itself, each line starts with a time stamp:

```
[27/Jun/2011:17:23:00 +0200] CONNECT conn=19 from=127.0.0.1:56641
 to=127.0.0.1:1389 protocol=LDAP
[27/Jun/2011:17:23:00 +0200] SEARCH REQ conn=19 op=0 msgID=1
 base="dc=example,dc=com" scope=wholeSubtree filter="(uid=bjensen)" attrs="ALL"
[27/Jun/2011:17:23:00 +0200] SEARCH RES conn=19 op=0 msgID=1
 result=0 nentries=1 etime=3
[27/Jun/2011:17:23:00 +0200] UNBIND REQ conn=19 op=1 msgID=2
[27/Jun/2011:17:23:00 +0200] DISCONNECT conn=19 reason="Client Unbind"
```

As you see, each client connection and set of LDAP operations are traced, starting with a time stamp and information about the operation performed, then including information about the connection, the operation number for the sequence of operations performed by the client, a message identification number, and additional information about the operation.

To match HTTP client operations with related internal server operations, first prevent OpenDJ from suppressing internal operations from the LDAP access log by using the `dsconfig` command to set the LDAP access log publisher `suppress-internal-operations` advanced property to `false`. Then match the values of the `x-connection-id` field in the HTTP access log with `conn=id` values in the LDAP access log.

For example, consider an HTTP GET request for the `_id` field of the user `newuser`, which is handled by connection 4 as shown in `logs/http-access`:

```
-  192.168.0.12  bjensen  22/May/2013:16:27:52 +0200
  GET  /users/newuser?_fields=_id  HTTP/1.1  200
  curl/7.21.4  4  12
```

With internal operations logged in `logs/access`, log lines for the related operations have `conn=4`:

```
[22/May/2013:16:27:52 +0200] CONNECT conn=4
  from=192.168.0.12:63593 to=192.168.0.12:8080 protocol=HTTP/1.1
[22/May/2013:16:27:52 +0200] SEARCH REQ conn=4
  op=0 msgID=0 base="ou=people,dc=example,dc=com" scope=wholeSubtree
   filter="(&(objectClass=inetOrgPerson)(uid=bjensen))" attrs="1.1"
[22/May/2013:16:27:52 +0200] SEARCH RES conn=4
  op=0 msgID=0 result=0 nentries=1 etime=5
[22/May/2013:16:27:52 +0200] BIND REQ conn=4
```

```
  op=1 msgID=1 version=3 type=SIMPLE
   dn="uid=bjensen,ou=People,dc=example,dc=com"
[22/May/2013:16:27:52 +0200] BIND RES conn=4
  op=1 msgID=1 result=0 authDN="uid=bjensen,ou=People,dc=example,dc=com"
   etime=3
[22/May/2013:16:27:52 +0200] SEARCH REQ conn=4
  op=2 msgID=2 base="uid=newuser,ou=people,dc=example,dc=com" scope=baseObject
   filter="(objectClass=*)" attrs="uid,etag"
[22/May/2013:16:27:52 +0200] SEARCH RES conn=4
   op=2 msgID=2 result=0 nentries=1 etime=4
[22/May/2013:16:27:52 +0200] UNBIND REQ conn=4
   op=3 msgID=3
[22/May/2013:16:27:52 +0200] DISCONNECT conn=4
   reason="Client Unbind"
```

To help diagnose errors due to access permissions, OpenDJ supports the get effective rights control. The control OID, `1.3.6.1.4.1.42.2.27.9.5.2`, is not allowed by the default global ACIs. You must therefore add access to use the get effective rights control when not using it as Directory Manager.

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, OpenDJ includes a global ACI to allow only authenticated users to use the control:

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

Property    : Value(s)
-----------:---------------------------------------------------------------------
enabled     : true
global-aci : (extop="1.3.6.1.4.1.26027.1.6.1 || 1.3.6.1.4.1.26027.1.6.3 ||
...
            : (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";), (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") (version 3.0; acl "Anonymous control
             : access"; allow(read) userdn="ldap:///anyone";)
```

To grant anonymous (unauthenticated) user access to the control, add the OID for the simple paged results control to the list of those in the `Anonymous control access` global ACI:

```
$ dsconfig \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword "password" \
 set-access-control-handler-prop \
 --remove global-aci:"(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\") (version 3.0; acl \"Anonymous control access\"; \
 allow(read) userdn=\"ldap:///anyone\";)" \
 --add global-aci:"(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 || 1.2.840.113556.1.4.319\") \
 (version 3.0; acl \"Anonymous control access\"; allow(read) \
 userdn=\"ldap:///anyone\";)" \
 --no-prompt
```

Alternatively, stop OpenDJ, edit the corresponding ACI carefully in `/path/to/opendj/config/config.ldif`, and restart OpenDJ. [1]

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:

```
[27/Jun/2011:14:37:48 +0200] category=SYNC severity=INFORMATION msgID=14680169
 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 maintains 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, OpenDJ purges 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 to you, disable the replica, and then reinitialize it from a recent backup or from a server that is up to date.

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 also the version of OpenDJ you use to reproduce the problem. The version includes both a version number and also a build time stamp:

```
$ dsconfig --version
OpenDJ 4.9.2
Build yyyymmddhhmmssZ
```

Be ready to provide the following additional information:

- The output from the `java -version` command.
- `access` and `errors` 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 OpenDJ 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.

[1] Unlike the`dsconfig`command, the`config.ldif`file is not a public interface, so this alternative should not be used in production.