Red Hat Directory Server Red Hat Directory Server 9

Administration Guide

Updated for Directory Server 9.1.2

Last Updated: 2020-10-30

Red Hat Directory Server Red Hat Directory Server 9 Administration Guide

Updated for Directory Server 9.1.2

Marc Muehlfeld
Red Hat Customer Content Services
mmuehlfeld@redhat.com

Petr Bokoč
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

Legal Notice

Abstract

This guide covers both GUI and command-line procedures for managing Directory Server instances and databases. This documentation is no longer maintained. For details, see .

DEPRECATED DOCUMENTATION

IMPORTANT

Note that as of June 10, 2017, the support for Red Hat Directory Server 9 has ended. For details, see "Red Hat Directory Server Life Cycle policy". Red Hat recommends users of Directory Server 9 to update to the latest version.

Due to the end of the maintenance phase of this product, this documentation is no longer updated. Use it only as a reference!

PREFACE

Red Hat Directory Server (Directory Server) is a powerful and scalable distributed directory server based on the industry-standard Lightweight Directory Access Protocol (LDAP). Directory Server is the cornerstone for building a centralized and distributed data repository that can be used in your intranet, over your extranet with your trading partners, or over the public Internet to reach your customers.

This *Administrator's Guide* describes all of the administration tasks you need to perform to maintain Directory Server.

1. DIRECTORY SERVER OVERVIEW

Directory Server provides the following key features:

- Multi-master replication — Provides a highly available directory service for both read and write operations. Multi-master replication can be combined with simple and cascading replication scenarios to provide a highly flexible and scalable replication environment.
- Chaining and referrals — Increases the power of your directory by storing a complete logical view of your directory on a single server while maintaining data on a large number of Directory Servers transparently for clients.
- Roles and classes of service — Provides a flexible mechanism for grouping and sharing attributes between entries in a dynamic fashion.
- Improved access control mechanisms — Provides support for macros that dramatically reduce the number of access control statements used in the directory and increase the scalability of access control evaluation.
- Resource-limits by bind DN — Grants the power to control the amount of server resources allocated to search operations based on the bind DN of the client.
- Multiple databases — Provides a simple way of breaking down your directory data to simplify the implementation of replication and chaining in your directory service.
- Password policy and account lockout — Defines a set of rules that govern how passwords and user accounts are managed in the Directory Server.
- TLS and SSL — Provides secure authentication and communication over the network, using the Mozilla Network Security Services (NSS) libraries for cryptography.

The major components of Directory Server include the following:

- An LDAP server — The LDAP v3-compliant network daemon.
- Directory Server Console — A graphical management console that dramatically reduces the effort of setting up and maintaining your directory service.
- SNMP agent — Can monitor the Directory Server using the Simple Network Management Protocol (SNMP).

2. EXAMPLES AND FORMATTING

Each of the examples used in this guide, such as file locations and commands, have certain defined conventions.

2.1. Command and File Examples

All of the examples for Red Hat Directory Server commands, file locations, and other usage are given for Red Hat Enterprise Linux 6 (64-bit) systems. Be certain to use the appropriate commands and files for your platform.

> **Example 1. Example Command**
>
> To start the Red Hat Directory Server:
>
> ```
> service dirsrv start
> ```

2.2. Brackets

Square brackets (**[]**) are used to indicate an alternative element in a name. For example, if a tool is available in **/usr/lib** on 32-bit systems and in **/usr/lib64** on 64-bit systems, then the tool location may be represented as **/usr/lib[64]**.

2.3. Client Tool Information

The tools for Red Hat Directory Server are located in the **/usr/bin** and the **/usr/sbin** directories.

IMPORTANT

The LDAP tools such as **ldapmodify** and **ldapsearch** from OpenLDAP use SASL connections by default. To perform a simple bind using a user name and password, use the **-x** argument to disable SASL.

2.4. Text Formatting and Styles

Certain words are represented in different fonts, styles, and weights. Different character formatting is used to indicate the function or purpose of the phrase being highlighted.

Formatting Style	Purpose
Monospace font	Monospace is used for commands, package names, files and directory paths, and any text displayed in a prompt.
`Monospace with a background`	This type of formatting is used for anything entered or returned in a command prompt.
Italicized text	Any text which is italicized is a variable, such as *instance_name* or *hostname*. Occasionally, this is also used to emphasize a new term or other phrase.
Bolded text	Most phrases which are in bold are application names, such as **Cygwin**, or are fields or options in a user interface, such as a **User Name Here:** field or **Save** button.

Other formatting styles draw attention to important text.

NOTE

A note provides additional information that can help illustrate the behavior of the system or provide more detail for a specific issue.

IMPORTANT

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.

WARNING

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

3. ADDITIONAL READING

The *Red Hat Directory Server Deployment Guide* describes many of the basic directory and architectural concepts that you need to deploy, install, and administer a directory service successfully.

When you are familiar with Directory Server concepts and have done some preliminary planning for your directory service, install the Directory Server. The instructions for installing the various Directory Server components are contained in the *Red Hat Directory Server Installation Guide* . Many of the scripts and commands used to install and administer the Directory Server are explained in detail in the *Red Hat Directory Server Configuration and Command-Line Tool Reference*.

The *Directory Server Administrator's Guide* describes how to set up, configure, and administer Red Hat Directory Server and its contents.

The document set for Directory Server contains the following guides:

- *Red Hat Directory Server Release Notes* contain important information on new features, fixed bugs, known issues and workarounds, and other important deployment information for this specific version of Directory Server.
- *Red Hat Directory Server Deployment Guide* provides an overview for planning a deployment of the Directory Server.
- *Red Hat Directory Server Administrator's Guide* contains procedures for the day-to-day maintenance of the directory service. Includes information on configuring server-side plug-ins.
- *Red Hat Directory Server Configuration and Command-Line Tool Reference* provides reference information on the command-line scripts, configuration attributes, schema elements, and log files shipped with Directory Server.
- *Red Hat Directory Server Installation Guide* contains procedures for installing your Directory Server as well as procedures for migrating from a previous installation of Directory Server.
- *Red Hat Directory Server Plug-in Programmer's Guide* describes how to write server plug-ins in order to customize and extend the capabilities of Directory Server.
- The *Red Hat Directory Server Performance Tuning Guide* contains features to monitor overall Directory Server and database performance, to tune attributes for specific operations, and to tune the server and database for optimum performance.

For the latest information about Directory Server, including current release notes, complete product documentation, technical notes, and deployment information, see the Red Hat Directory Server documentation site at https://access.redhat.com/site/documentation/Red_Hat_Directory_Server/.

4. GIVING FEEDBACK

If there is any error in this *Administrator's Guide* or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for Red Hat Directory Server through Bugzilla, http://bugzilla.redhat.com/bugzilla. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

1. Select the Red Hat Directory Server product.
2. Set the component to **Doc - administration-guide**.
3. Set the version number to 9.0.
4. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.

   For enhancements, put in what information needs to be added and why.

5. Give a clear title for the bug. For example, **"Incorrect command example for setup script options"** is better than **"Bad example"**.

We appreciate receiving any feedback – requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at docs@redhat.com.

CHAPTER 1. BASIC RED HAT DIRECTORY SERVER SETTINGS

Red Hat Directory Server product includes a directory service, an administration server to manage multiple server instances, and a Java-based console to manage server instances through a graphical interface. This chapter provides an overview of the basic tasks for administering a directory service.

The Directory Server is a robust, scalable server designed to manage an enterprise-wide directory of users and resources. It is based on an open-systems server protocol called the Lightweight Directory Access Protocol (LDAP). Directory Server runs the **ns-slapd** daemon on the host machine. The server manages the directory databases and responds to client requests.

Directory Server 9.0 is comprised of several components, which work in tandem:

- The *Directory Server* is the core LDAP server daemon. It is compliant with LDAP v3 standards. This component includes command-line server management and administration programs and scripts for common operations like export and backing up databases.
- The *Directory Server Console* is the user interface that simplifies managing users, groups, and other LDAP data for your enterprise. The Console is used for all aspects of server management, including making backups; configuring security, replication, and databases; adding entries; and monitoring servers and viewing statistics.
- The *Admin Server* is the management agent which administers Directory Server instances. It communicates with the Directory Server Console and performs operations on the Directory Server instances. It also provides a simple HTML interface and online help pages.

Most Directory Server administrative tasks are available through the Directory Server Console, but it is also possible to administer the Directory Server by manually editing the configuration files or by using command-line utilities.

1.1. SYSTEM REQUIREMENTS

This section contains information related to installing and upgrading Red Hat Directory Server 9.0, including prerequisites and hardware or platform requirements.

1.1.1. Required JRE

Red Hat Directory Server 9.0 requires the Oracle Java Runtime Environment (JRE) 1.8.0 or OpenJDK 1.8.0 for Red Hat Enterprise Linux 5 and 6.

IMPORTANT

It is not possible to manage instances of Directory Server older than 8.1 (which used JRE 1.5) with the 9.0 Directory Server Console because they are using different JRE versions.

1.1.2. Directory Server Supported Platforms

Directory Server 9.0 is supported on the following platforms:

- Red Hat Enterprise Linux 6 (32-bit)
- Red Hat Enterprise Linux 6 (64-bit)

NOTE

Red Hat Directory Server 9.0 is supported running on a virtual guest on a Red Hat Enterprise Linux virtual server.

1.1.3. Directory Server Console Supported Platforms

The Directory Server Console is supported on the following platforms:

- Red Hat Enterprise Linux 5 (32-bit)
- Red Hat Enterprise Linux 5 (64-bit)
- Red Hat Enterprise Linux 6 (32-bit)
- Red Hat Enterprise Linux 6 (64-bit)
- Microsoft Windows Server 2008 (32-bit)

- Microsoft Windows Server 2008 (64-bit)

1.1.4. Password Sync Service Platforms

The Password Sync Service works with these Microsoft Windows services:

- Active Directory on Microsoft Windows Server 2008 (32-bit)
- Active Directory on Microsoft Windows Server 2008 (64-bit)

1.1.5. Web Application Browser Support

Directory Server 9.0 supports the following browsers to access web-based interfaces, such as **Admin Express** and online help tools:

- Firefox 3.x
- Microsoft Internet Explorer 6.0 and higher

1.2. DIRECTORY SERVER FILE LOCATIONS

Red Hat Directory Server 9.0 conforms to the Filesystem Hierarchy Standards. For more information on FHS, see the FHS homepage, http://www.pathname.com/fhs/. The files and directories installed with Directory Server are listed in the tables below for each supported platform.

In the file locations listed in the following tables, *instance* is the server instance name that was given during setup. By default, this is the leftmost component of the fully-qualified host and domain name. For example, if the host name is **ldap.example.com**, the instance name is **ldap** by default.

The Admin Server directories are named the same as the Directory Server directories, only instead of the instance as a directory name, the Admin Server directories are named **admin-serv**. For any directory or folder named **slapd-***instance*, substitute **admin-serv**, such as **/etc/dirsrv/slapd-example** and **/etc/dirsrv/admin-serv**.

Table 1.1. Red Hat Enterprise Linux 5 (x86)

File or Directory	Location
Log files	**/var/log/dirsrv/slapd-***instance*
Configuration files	**/etc/dirsrv/slapd-***instance*
Instance directory	**/usr/lib/dirsrv/slapd-***instance*
Certificate and key databases	/etc/dirsrv/slapd-*instance*
Database files	**/var/lib/dirsrv/slapd-***instance*
Runtime files	**/var/lock/dirsrv/slapd-***instance* **/var/run/dirsrv/slapd-***instance*
Initscripts	**/etc/rc.d/init.d/dirsrv** and **/etc/sysconfig/dirsrv** **/etc/rc.d/init.d/dirsrv-admin** and **/etc/sysconfig/dirsrv-admin**
Tools	**/usr/bin/** **/usr/sbin/**

Table 1.2. Red Hat Enterprise Linux 5 and 6 (x86_64)

File or Directory	Location
Log files	**/var/log/dirsrv/slapd-***instance*

File or Directory	Location
Configuration files	**/etc/dirsrv/slapd-***instance*
Instance directory	**/usr/lib64/dirsrv/slapd-***instance*
Certificate and key databases	/etc/dirsrv/slapd-*instance*
Database files	**/var/lib/dirsrv/slapd-***instance*
Runtime files	**/var/lock/dirsrv/slapd-***instance* **/var/run/dirsrv/slapd-***instance*
Initscripts	**/etc/rc.d/init.d/dirsrv** and **/etc/sysconfig/dirsrv** **/etc/rc.d/init.d/dirsrv-admin** and **/etc/sysconfig/dirsrv-admin**
Tools	**/usr/bin/** **/usr/sbin/**

1.3. STARTING AND STOPPING SERVERS

The Directory Server is running when the **setup-ds-admin.pl** script completes. Avoid stopping and starting the server to prevent interrupting replication, searches, and other server operations.

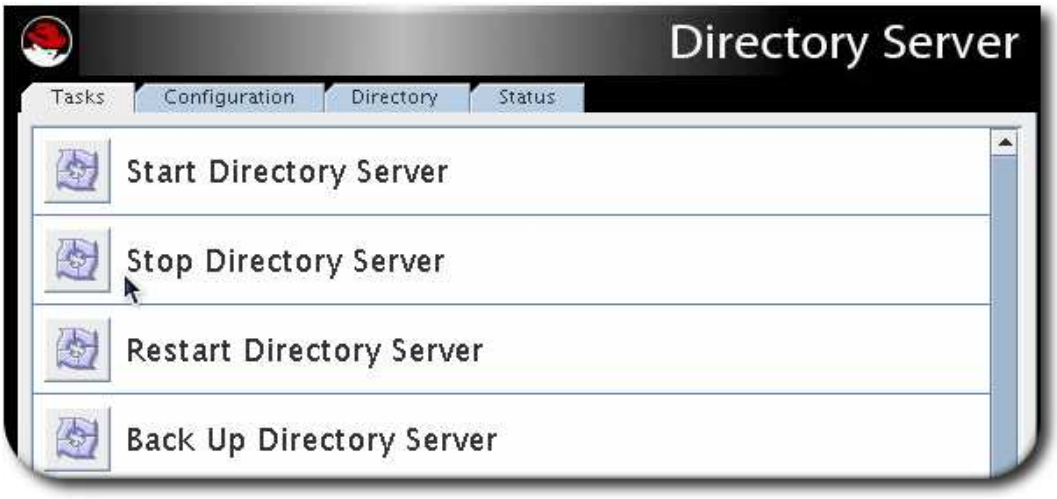
- If the Directory Server has SSL enabled, you cannot restart the server from the Console; you must use the command-line. It is possible to restart without being prompted for a password; see Section 7.4.4, "Creating a Password File for the Directory Server" for more information.
- Rebooting the host system can automatically start the **ns-slapd** process. The directory provides startup or run command (**rc**) scripts. Use the **chkconfig** command to enable the Directory Server and Admin Server to start on boot.

1.3.1. Starting and Stopping Directory Server from the Console

1. Start the Directory Server Console.

```
redhat-idm-console -a http://localhost:9830
```

2. In the **Tasks** tab, click **Start the Directory Server**, **Stop the Directory Server**, or **Restart the Directory Server**.



When the Directory Server is successfully started or stopped from the Directory Server Console, the server displays a message box stating that the server has either started or shut down.

1.3.2. Starting and Stopping Directory Server from the Command Line

The most common way to start and stop the Directory Server service is using system tools on Red Hat Enterprise Linux. For example, Linux uses the **service** tool:

```
service dirsrv {start|stop|restart} instance
```

Passing the instance name stops or starts only that instance; not giving any name starts or stops all instances.

NOTE

The service name for the Directory Server service on Red Hat Enterprise Linux is **dirsrv**.

The start/stop scripts are in the **/usr/sbin/** directory and are run similar to the **service** start/stop command:

```
/usr/sbin/{start|stop|restart}-dirsrv instance
```

If the instance name is not given, then the all instances are started or stopped.

Alternatively, each instance has its own start and stop scripts that apply only to that instance.

```
/etc/dirsrv/slapd-instance_name/{start|stop|restart}-slapd
```

1.3.3. Starting and Stopping Admin Server

The Admin Server service is stopped and started using system tools on Red Hat Enterprise Linux. For example, on Red Hat Enterprise Linux 6 (64-bit), the command is **service**:

```
service dirsrv-admin {start|stop|restart}
```

NOTE

The service name for the Admin Server process on Red Hat Enterprise Linux is **dirsrv-admin**.

1.4. STARTING THE CONSOLE

1.4.1. Starting the Directory Server Console

There is a simple script to launch the Directory Server Console. The script is in the standard **/usr/bin** directory, so it can be run as follows:

```
redhat-idm-console
```

NOTE

Make sure that the correct Oracle Java Runtime Environment (JRE) or OpenJDK version is set in the **PATH** variable before launching the Console. To display the version and vendor information about Java on your system, enter:

```
java -version
```

The login screen prompts for the user name, password, and Admin Server location. It is possible to pass other information along with the Console command to supply the Admin Server URL, password, and user name. For example:

```
redhat-idm-console -a http://localhost:9830 -u "cn=Directory Manager" -w secret
```

Table 1.3. redhat-idm-console Options

Option	Description
-a *adminURL*	Specifies a base URL for the instance of Admin Server to log into.

Option	Description
-f *fileName*	Writes errors and system messages to *fileName*.
-h	Prints out the help message for **redhat-idm-console**.
-s	Specifies the directory instance to access, either by specifying the DN of the server instance entry (SIE) or the instance name, such as **slapd-example**.
-u	Gives the user DN to use to log into the Console.
-w	Gives the password to use to log into the Console.
-w -	Reads the password from the standard output.
-x *options*	Specifies extra options. There are three values for *extraOptions*: nowinpos, which puts the Console window in the upper left corner of the screen nologo, which keeps the splash screen from being displayed and only opens the login dialog javalaf, which uses the *Java look and feel* for the Console interface rather than the platform-specific styles To use multiple options, separate them with a comma.
-y *file*	Reads the password from the specified input file.

1.4.2. Logging into Directory Server

After starting the Directory Server Console, a login screen opens, requiring the user name and password for the user logging in and the URL for the Admin Server instance being access. The user logged in at the Console is the user who is *binding* to Directory Server. This determines the access permissions granted and allowed operations while access the directory tree. The user account used to log into the Directory Server Console can make significant differences in the access; for example, the Directory Manager has access to every user and configuration entry in Directory Server, while the **admin** entry created during installation has access to only configuration entries, not user entries. Regular user accounts are more limited.

To bind to, or log into, the Directory Server, supply a user name and password at the login box.

1.4.3. Changing the Login Identity

At any time during a session, you can log in as a different user, without having to restart the Console. To change the login identity:

1. In the Directory Server Console, select the **Tasks** tab.
2. Click **Log on to the Directory Server as a New User**.

3. A login dialog box appears.

Enter the full distinguished name of the entry with which to bind to the server. For example, to bind as user Barbara Jensen, enter her full DN in the login box:

```
cn=Barbara Jensen,ou=People,dc=example,dc=com
```

1.4.4. Viewing the Current Console Bind DN

To see the bind DN that is currently logged into the Directory Server Console, click the login icon in the lower-left corner of the window. The current bind DN appears next to the login icon.

Figure 1.1. Viewing the Bind DN

1.5. ENABLING LDAPI

Inter-process communication (IPC) is a way for separate processes on a Unix machine or a network to communicate directly with each other. LDAPI allows LDAP connections to run over IPC connections, meaning that LDAP operations can run over Unix sockets. These connections are much faster and more secure than regular LDAP connections.

LDAPI is enabled through two configuration attributes:

- ***nsslapd-ldapilisten*** to enable LDAPI for Directory Server
- ***nsslapd-ldapifilepath*** to point to the Unix socket file

To enable LDAPI:

1. Modify the ***nsslapd-ldapilisten*** to turn LDAPI on and add the socket file attribute.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-ldapilisten
nsslapd-ldapilisten: on
-
add: nsslapd-ldapifilepath
nsslapd-ldapifilepath: /var/run/slapd-example.socket
```

2. Restart the server to apply the new configuration.

```
service dirsrv restart example
```

1.6. CHANGING DIRECTORY SERVER PORT NUMBERS

The standard and secure LDAP port numbers used by Directory Server can be changed through the Directory Server Console or by changing the value of the ***nsslapd-port*** or ***nsslapd-secureport*** attribute under the **cn=config** entry in the **dse.ldif**.

NOTE

Modifying the standard or secure port numbers for a Configuration Directory Server, which maintains the **o=NetscapeRoot** subtree, should be done through the Directory Server Console.

1.6.1. Changing Standard Port Numbers

1. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
2. Select the **Settings** tab in the right pane.

3. Change the port numbers. The port number for the server to use for non-SSL communications in the **Port** field, with a default value of **389**.
4. Click **Save**.

5. The Console returns a warning, *You are about to change the port number for the Configuration Directory. This will affect all Administration Servers that use this directory and you'll need to update them with the new port number. Are you sure you want to change the port number?* Click **Yes**.

6. Then a dialog appears, reading that the changes will not take effect until the server is restarted. Click **OK**.

NOTE

Do not restart the Directory Server at this point. If you do, you will not be able to make the necessary changes to the Admin Server through the Console.

7. Open the Admin Server Console.

8. In the **Configuration** tab, select the **Configuration DS** tab.

9. In the **LDAP Port** field, type in the new LDAP port number for your Directory Server instance.

10. Change the SELinux labels for the Directory Server ports so that the new port number is used in the Directory Server policies. By default, only port 389 is labeled. The process for labeling ports is covered in Section 1.10.6, "Labeling SSL/TLS Ports". For example:

```
/usr/sbin/semanage port -a -t ldap_port_t -p tcp 1389
```

WARNING

If the SELinux label is not reset, then the Directory Server will not be able to be restarted.

11. In the **Tasks** tab of the Directory Server Console, click **Restart Directory Server**. A dialog to confirm that you want to restart the server. Click **Yes**.

12. Open the **Configuration DS** tab of the Admin Server Console and select **Save**.

    A dialog will appear, reading *The Directory Server setting has been modified. You must shutdown and restart your Admin Server and all the servers in the Server Group for the changes to take effect.* Click **OK**.

13. In the **Tasks** tab of the Admin Server Console, click **Restart Admin Server**. A dialog opens reading that the Admin Server has been successfully restarted. Click **Close**.

NOTE

You *must* close and reopen the Console before you can do anything else in the Console. Refresh may not update the Console, and, if you try to do anything, you will get a warning that reads *Unable to contact LDAP server.*

1.6.2. Changing SSL Port Numbers

Changing the configuration directory or user directory port or secure port numbers has the following repercussions:

- The Directory Server port number must also be updated in the Admin Server configuration.
- If there are other Directory Server instances that point to the configuration or user directory, update those servers to point to the new port number.

To modify the LDAPS port:

1. Make sure that the CA certificate used to issue the Directory Server instance's certificate is in the Admin Server certificate database. Importing CA certificates for the Admin Server is the same as the Directory Server process described in Section 7.3.2, “Trusting the Certificate Authority” .

2. The secure port can be configured using the Directory Server Console, much like the process in Section 1.6.1, “Changing Standard Port Numbers” (only setting the value in the **Encrypted Port** field). However, in some circumstances, such as if there are multiple Directory Server instances on the same machine, where changing port numbers may not be possible through the Directory Server Console. It may be be better to use **ldapmodify** to change the port number.

   For example:

```
[root@server ~]# ldapmodify -x -h server.example.com -p 1389 -D "cn=directory manager" -W
dn: cn=config
replace: nsslapd-securePort
nsslapd-securePort: 1636
```

3. Edit the corresponding port configuration for the Directory Server instance in th Admin Server configuration (**o=netscaperoot**).

   First, search for the current configuration:

```
[root@server ~]# ldapsearch -x -h config-ds.example.com -p 389 -D "cn=directory manager" -W -b
"cn=slapd-ID,cn=389 Directory Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot" -s base "(objectclass=*)"
```

```
nsSecureServerPort

dn: cn=slapd-ID,cn=389 Directory Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
nsSecureServerPort: 636
```

Then, edit the configuration:

```
[root@server ~]# ldapmodify -x -h config-ds.example.com -p 389 -D "cn=directory manager" -WW

dn: cn=slapd-ID,cn=389 Directory Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
replace: nsSecureServerPort
nsSecureServerPort: 1636
```

4. Start the Directory Server Console for the instance and confirm that the new SSL port number is listed in the **Configuration** tab.

5. Optionally, select the **Use SSL in Console** check box.

6. Change the SELinux labels for the Directory Server ports so that the new port number is used in the Directory Server policies. By default, only port 389 is labeled. The process for labeling ports is covered in Section 1.10.6, "Labeling SSL/TLS Ports". For example:

```
/usr/sbin/semanage port -a -t ldap_port_t -p tcp 1636
```

WARNING

If the SELinux label is not reset, then the Directory Server will not be able to be restarted.

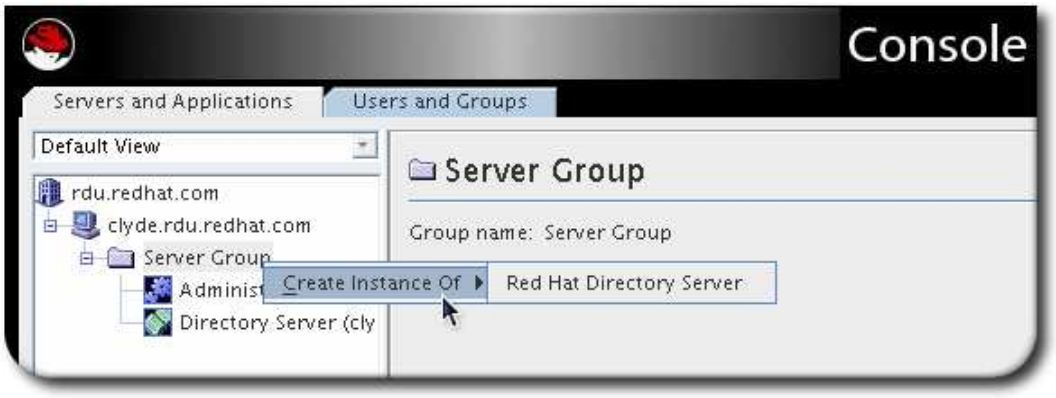
7. Restart the Directory Server instance.

1.7. CREATING A NEW DIRECTORY SERVER INSTANCE

Additional instances can be created through the Directory Server Console or using the **setup-ds.pl** script. For information on using the **setup-ds.pl** script, see the *Directory Server Installation Guide*. To create an instance using the Directory Server Console:

1. In the Red Hat Console window, select **Server Group** in the navigation tree, and then right-click.

2. From the pop-up menu, select **Create Instance** and then **Directory Server**.



3. Fill in the instance information.

- A unique name for the server. This name must only have alphanumeric characters, a dash (**-**), or an underscore (_).
- A port number for LDAP communications.
- The root suffix for the new Directory Server instance.
- A DN for the Directory Manager. This user has total access to every entry in the directory, without normal usage constraints (such as search timeouts).
- The password for the Directory Manager.
- The user ID as which to run the Directory Server daemon.

4. Click **OK**.

A status box appears to confirm that the operation was successful. To dismiss it, click **OK**.

1.8. USING DIRECTORY SERVER PLUG-INS

Directory Server has a number of default plug-ins which configure core Directory Server functions, such as replication, classes of service, and even attribute syntaxes. Core plug-ins are enabled and completely configured by default.

Other default plug-ins extend the functionality of the Directory Server by providing consistent, but user-defined, behaviors, as with DNA, attribute uniqueness, and attribute linking. These plug-ins are available, but not enabled or configured by default.

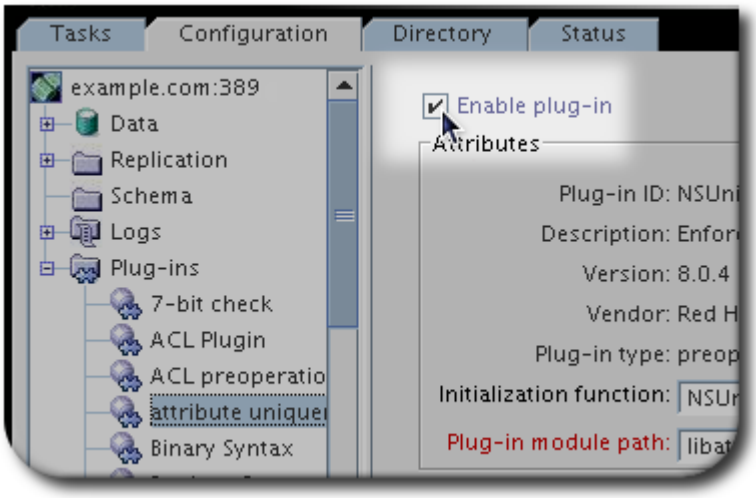
Using plug-ins also allows the Directory Server to be easily extended, so customers can write and deploy their own server plug-ins to perform whatever directory operations they need for their specific deployment.

The details of configuring and deploying plug-ins are covered in other guides (primarily the *Plug-in Programmer's Guide* and to some extent in the plug-in attribute reference in the *Configuration and Command-Line Tool Reference*). This section covers common administrative tasks for all plug-ins.

1.8.1. Enabling Plug-ins in the Directory Server Console

To enable and disable plug-ins over LDAP using the Directory Server Console:

1. In the Directory Server Console, select the **Configuration** tab.
2. Double-click the **Plugins** folder in the navigation tree.
3. Select the plug-in from the **Plugins** list.
4. To disable the plug-in, clear the **Enabled** check box. To enable the plug-in, check this check box.



5. Click **Save**.

6. Restart the Directory Server.

```
service dirsrv restart instance_name
```

NOTE

When a plug-in is disabled, all of the details about the plug-in — such as its version and its vendor — are not displayed in the Directory Server Console; all details fields show **NONE**.

Once a plug-in is enabled, those details will not be displayed in the Console until the Directory Server is restarted (loading the new plug-in configuration) and the Directory Server Console is refreshed.

1.8.2. Enabling Plug-ins in the Command Line

To disable or enable a plug-in through the command line, use the **ldapmodify** utility to edit the value of the ***nsslapd-pluginEnabled*** attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ACL Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

1.8.3. Setting the Plug-in Precedence

The plug-in precedence is the priority it has in the execution order of plug-ins. For pre- and post-operation plug-ins, this allows one plug-in to be executed and complete before the next plug-in is initiated, which lets the second plug-in take advantage of the first plug-in's results.

Plug-in precedence is configured in the ***nsslapd-pluginPrecedence*** attribute on the plug-in's configuration entry. This attribute has a value of 1 (highest priority) to 99 (lowest priority). If the attribute is not set, it has a default value of 50.

The ***nsslapd-pluginPrecedence*** attribute is set using the **ldapmodify** command. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=My Example Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginPrecedence
nsslapd-pluginPrecedence: 1
```

IMPORTANT

Don not set the plug-in precedence for the default Directory Server plug-ins unless told to do so by Red Hat support. The plug-in precedence attribute is primarily to govern the behavior of *custom* plug-ins, not to change the behavior of the core Directory Server plug-ins.

1.9. MANAGING CORE SERVER ATTRIBUTES

The Directory Server configuration itself is stored in the **dse.ldif** file, which contains the server configuration entries like **cn=config**. The server entry itself is defined through a finite and strict set of attributes called *core server configuration attributes*. Although these attributes can be changed, no attributes can be added to the core server configuration and none can be deleted (except under very limited circumstances, as described in Section 1.9.2, "Configuration Attributes Which Can Be Deleted").

This is described in more detail in the overview sections of the "Server Instance File Reference" chapter in the *Directory Server Configuration and Command-Line Tool Reference* .

This section provides details on how to check which core server attributes require that the server be restarted and how to check or change which core server configuration attributes can be deleted.

1.9.1. Configuration Attributes Requiring Server Restart

Some configuration attributes cannot be altered while the server is running. In these cases, for the changes to take effect, the server needs to be shut down and restarted. The modifications should be made either through the Directory Server Console or by manually editing the **dse.ldif** file when the **dirsrv** service is stopped.

Some of the attributes that require a server restart for any changes to take effect are listed below. This list is not exhaustive; to see a complete list, run **ldapsearch** and search for the ***nsslapd-requiresrestart*** attribute. For example:

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "cn=config" -s sub -x "(objectclass=*)"
| grep nsslapd-requiresrestart
```

nsslapd-cachesize	***nsslapd-certdir***	***nsslapd-dbcachesize***
nsslapd-dbncache	***nsslapd-plugin***	***nsslapd-changelogdir***
nsslapd-changelogmaxage	***nsslapd-changelogmaxentries***	***nsslapd-port***
nsslapd-schemadir	***nsslapd-saslpath***	***nsslapd-secureport***
nsslapd-tmpdir	***nsSSL2***	***nsSSL3***
nsTLS1	***nsSSLclientauth***	***nsSSLSessionTimeout***
nsslapd-conntablesize	***nsslapd-lockdir***	***nsslapd-maxdescriptors***
nsslapd-reservedescriptors	***nsslapd-listenhost***	***nsslapd-schema-ignore-trailing-spaces***
nsslapd-securelistenhost	***nsslapd-workingdir***	***nsslapd-return-exact-case***
nsslapd-maxbersize[a]	***nsslapd-allowed-to-delete-attrs***	

[a] Although this attribute requires a restart, it is not returned in the search.

1.9.2. Configuration Attributes Which Can Be Deleted

Core server configuration attributes cannot be deleted, by default. All core configuration attributes are present, even if they are not written in the **dse.ldif** file, because they all have default values used by the server. Deleting any of those attributes is generally not allowed because the server requires that those attributes be present for it to run.

The ***nsslapd-allowed-to-delete-attrs*** parameter lists core configuration attributes which are *allowed* to be deleted from the configuration. Delete operations for those attributes will succeed.

The value of ***nsslapd-allowed-to-delete-attrs*** is a space-separated list of attribute names. By default, only two attributes are listed:

```
nsslapd-allowed-to-delete-attrs: nsslapd-listenhost nsslapd-securelistenhost
```

This can be changed using **ldapmodify** to add attributes to the list. Since this is a single-valued attribute, the *entire* list must be given in the modify statement; the modify operation overwrites the previous value, it does not append new values to it.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-allowed-to-delete-attrs
nsslapd-allowed-to-delete-attrs: nsslapd-listenhost nsslapd-securelistenhost nsslapd-rewrite-rfc1274
```

WARNING

Be extremely cautious about adding core server configuration attributes to the list of deletable attributes. Some attributes are critical for the server to operate, and deleting those attributes could cause the server not to run.

To return the list of attributes which can be deleted, use grep:

```
# egrep nsslapd-allowed-to-delete-attrs /etc/dirsrv/slapd-instance_name/dse.ldif

nsslapd-allowed-to-delete-attrs: nsslapd-listenhost nsslapd-securelistenhost nsslapd-rewrite-rfc1274
```

1.10. MANAGING SELINUX WITH THE DIRECTORY SERVER

SELinux is a security function in Linux that categorizes files, directories, ports, processes, users, and other objects on the server. Each object is placed in an appropriate security context to define how the object is allowed to behave on the server through its role, user, and security level. These roles for objects are grouped in domains, and SELinux rules define how the objects in one domain are allowed to interact with objects in another domain.

SELinux itself is much more complex to manage and implement than what is described here. This section is concerned only with giving the SELinux details for the Directory Server. Both the Fedora project and the National Security Agency have excellent resources for learning about SELinux.

NOTE

SELinux is a feature of Red Hat Enterprise Linux and, as such, is covered in the Red Hat Enterprise Linux *SELinux Guide* at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html.

1.10.1. SELinux Definitions for the Directory Server

SELinux has three different levels of enforcement: disabled (no SELinux), permissive (where the rules are lax), and enforcing (where all rules are strictly enforced). Red Hat Directory Server has defined SELinux policies that allow it to run as normal under strict SELinux enforcing mode, with a caveat. The Directory Server can run in different modes, one for normal operations and one for database operations like importing (ldif2db mode). The SELinux policies for the Directory Server only apply to normal mode.

By default, the Directory Server runs confined by SELinux policies.

The Directory Server processes are contained within the **dirsrv_t** domain. Ports used by the Directory Server instances are contained within the **ldap_port_t**domain.

Table 1.4, “Summary of Directory Server SELinux Policies” lists the security contexts and domains for the major components of the Directory Server.

Table 1.4. Summary of Directory Server SELinux Policies

File Path	Security Context	Description
dirsrv_t Domain		
/etc/dirsrv/*	dirsrv_config_t	Configuration files for the different instances.
/usr/sbin/ns-slapd	dirsrv_exec_t	The main server executable.
/usr/sbin/{start|restart|stop}-dirsrv	initrc_exec_t	The server start, restart, and stop scripts.
/usr/lib/dirsrv/* **/usr/lib64/dirsrv/***	lib_t	The server and plug-in libraries.
/usr/share/dirsrv/*	dirsrv_share_t	The property files and templates for new instances.
/var/lib/dirsrv/*	dirsrv_var_lib_t	The default directories for database files, LDIF files, and backup files.
/var/lock/dirsrv/*	dirsrv_var_lock_t	Lock files.
/var/log/dirsrv/*	dirsrv_var_log_t	The server instance log files.
/var/run/dirsrv/*	dirsrv_var_run_t	The instance PID files and the SNMP statistics file.
ldap_t Domain		
Port 389 and 636 and any regular LDAP port configured for a Directory Server instance	ldap_port_t	The ports used by the Directory Server instances, including the default LDAP and LDAPS ports and whatever the configured LDAP port[a] for the Directory Server is

[a] Only the LDAP port is configured for the Directory Server when it is set up, so only this port is added to the SELinux configuration automatically. The LDAPS port must be added manually, as described in Section 1.10.6, "Labeling SSL/TLS Ports".

The Directory Server SELinux policies are configured when the server instance is set up (when **setup-ds-admin.pl** or **register-ds-admin.pl** are run). Each time a new instance is configured, the policies are updated with the appropriate information. These policies are automatically removed when the server instance is uninstalled.

1.10.2. SELinux Definitions for the SNMP Agent

The Directory Server runs an SNMP agent which can be used to configure traps and send alerts to an SNMP master agent, as described in Chapter 16, *Monitoring Directory Server Using SNMP* . The SNMP sub-agent is contained within a separate domain, **dirsrv_snmp_t**.

The SNMP subagent runs as a process, **ldap-agent**. The process does not listen over any ports (the third-party SNMP master agent does), but the process does need to access some system files, such as PID and log files. The security context definitions for these files and process are listed in Table 1.5, "Summary of Directory Server SELinux Policies" . All of these files are also covered by the Directory Server file contexts listed in Table 1.4, "Summary of Directory Server SELinux Policies".

Table 1.5. Summary of Directory Server SELinux Policies

File Path	Security Context	Description

File Path	Security Context	Description
dirsrv_snmp_t Domain		
/usr/sbin/ldap-agent-bin	dirsrv_snmp_exec_t	The SNMP subagent daemon.
/var/run/ldap-agent.pid	dirsrv_snmp_var_run_t	The SNMP subagent PID file.
/var/log/dirsrv/ldap-agent.log	dirsrv_snmp_var_log_t	The SNMP subagent log file.

1.10.3. Viewing and Editing SELinux Policies for the Directory Server

The configured Directory Server and Admin Server policies can be viewed and edited using the SELinux Administration GUI. Much more information about editing SELinux policies and labels is in the Red Hat Enterprise Linux Security-Enhanced Linux Guide.

1. Open the **Systems** menu.
2. Open the **Administration** menu, and select the **SELinux Management** item.

NOTE

You can launch the GUI from the command line using **system-config-selinux**.

3. Open, add, or edit any file or port label or policy for Directory Server , as necessary.
4. After making any changes to the SELinux policies, run **restorecon** to load the changes to the labels or policies.

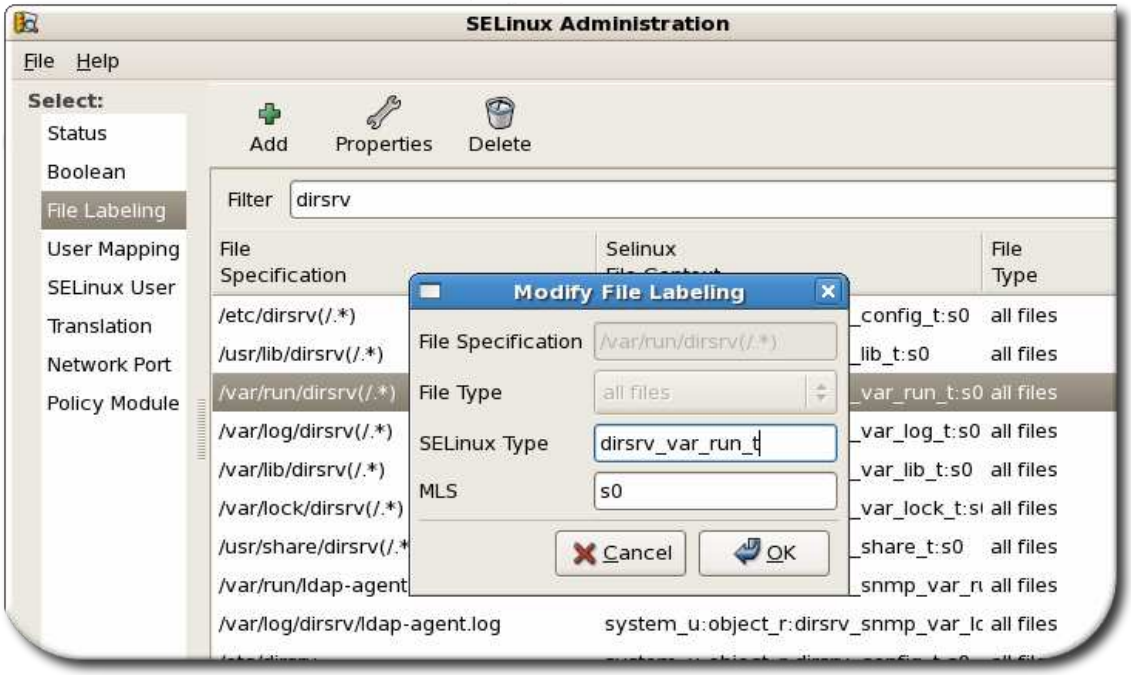
```
# restorecon -r -v [-f filename | directoryName]
```

For example, if new policies were created for a custom LDIF directory:

```
# restorecon -r -v /myNewLdifDir
```

To check the version of the Directory Server SELinux policy installed, click the **Policy Module** link.

To view the policies set on the individual files and processes, click the **File Labeling** link. To view the policies for the port assignments for the server, click the **Network Port** link.



1.10.4. Starting the Directory Server Confined by SELinux

Three scripts control how the **ns-slapd** process transitions to the **dirsrv_t** domain when starting and stopping. All three of these scripts are in the **/usr/sbin/** directory:

- start-dirsrv
- stop-dirsrv
- restart-dirsrv

These scripts are run similar to the **service** commands used by Directory Server. A single instance can be specified using the instance name or the script can be run with no arguments and apply to all instances, as in Section 1.3, "Starting and Stopping Servers". For example:

```
/usr/sbin/start-dirsrv instance_name
```

Likewise, the SNMP subagent is started or stopped using the **service** command to run the **ldap-agent** process confined by SELinux policies. See Section 16.3.2, "Starting the Subagent" for more information.

```
service dirsrv-snmp start
```

1.10.5. Managing SELinux Labels for Files Used by the Directory Server

There are a number of different files that the Directory Server has to access in normal operations, such as database, log, and index files. Many of these are configured in settings in **cn=config**, such as ***nsslapd-dbdir***, ***nsslapd-rundir***, and ***nsslapd-ldapifilepath***. As long as these directory locations are left with their default settings, the confined **ns-slapd** process can access them just fine. However, if these file locations are moved, then the SELinux labels must be updated for the new locations so that the Directory Server process is allowed to access them.

NOTE

Do not change the default locations for Directory Server files and directories — such as the databases, run file, or LDAPI configuration file — so that the SELinux policies do not have to be updated.

Most common files used by the Directory Server are covered by the SELinux policies by default. However, for some operations, the Directory Server must access *external* files, meaning files not directly created from Directory Server templates and maintained by the server. For example:

- *LDIF files for import and export* . If the import or export LDIF files are created in the default LDIF directory, **/var/lib/dirsrv/slapd-*instance_name*/ldif**, then the files will automatically be covered by the security context. If these are in a non-standard location, then the file labels must be changed for the Directory Server to access them.

  These SELinux labels apply only to the LDIF *files* used for import/export operations. These contexts do not cover import or export operations, which are database operations and outside the purview of SELinux.

  IMPORTANT

  If you copy a file into the LDIF directory, then the command automatically relabels the copied files and everything is fine. If, however, a file is moved into the LDIF directory (**mv**), then it retains its original SELinux labels and will not be recognized by the **ns-slapd** process.

- *Custom plug-ins*. The SELinux file restrictions assume that any plug-in files used by the server are located in the default plug-in directory, /usr/lib[64]/dirsrv/plugins on Red Hat Enterprise Linux 6 (64-bit). Any **.so** files for custom plug-ins must be in that directory for the server to load and use them.

  If the plug-in files must be stored in a non-default location for some reason, then add appropriate SELinux rules to allow the server to access the files. This is in Section 1.10.3, "Viewing and Editing SELinux Policies for the Directory Server" or using **semanage**.

- *SASL/GSS-API keytabs*. The Directory Server must be able to access the host keytab and **krb5.conf** configuration file for GSS-API authentication in SASL. (The host keytab is set in the **KRB5_KTNAME** directive in the **/etc/sysconfig/dirsrv** file.) For these files to be properly labeled in SELinux in the **dirsrv_config_t** context, they must be in the **/etc/dirsrv/** directory.

  Only the host keytab and **krb5.conf** file must be in **/etc**. The user key tabs can still be in any directory.

Although import/export operations and SASL configurations are the most common situations when the Directory Server will access an external file, be sure to consider file labeling any time the Directory Server needs to access a file.

File labels can be added using the SELinux administrative interface (Section 1.10.3, "Viewing and Editing SELinux Policies for the Directory Server") or using the **semanage** script. For details, see the semanage(8) man page.

1.10.6. Labeling SSL/TLS Ports

When the Directory Server is first set up, the given LDAP port is labeled for SELinux (the default is port 389). However, SSL/TLS is set up separately, after the Directory Server is already configured, so the LDAPS port for the Directory Server is not automatically labeled.

The default LDAP and LDAPS ports, 389 and 636, respectively, are already labeled as part of the policies in Red Hat Enterprise Linux. Any other LDAP port is added to those policies when the server is set up. If the Directory Server uses a secure port other than the defaults for its SSL/TLS connections, however, then an administrator must label the port manually. This can be done in the SELinux administrative interface shown in Section 1.10.3, "Viewing and Editing SELinux Policies for the Directory Server". It can also be done easily using the **semanage** script.

Use the **port** subcommand, the **-t** option to identify the security context, and the **-p** option to identify the port. The **-a** option adds the port label. For example:

```
/usr/sbin/semanage port -a -t ldap_port_t -p tcp 1636
```

To delete a port label, use the **-d** option. For example:

```
/usr/sbin/semanage port -d -t ldap_port_t -p tcp 1636
```

CHAPTER 2. CONFIGURING DIRECTORY DATABASES

The directory is made up of databases, and the directory tree is distributed across the databases. This chapter describes how to create *suffixes*, the branch points for the directory tree, and how to create the databases associated with each suffix. This chapter also describes how to create database links to reference databases on remote servers and how to use referrals to point clients to external sources of directory data.

For a discussion of concepts about distributing directory data, see the *Directory Server Deployment Guide*.

2.1. CREATING AND MAINTAINING SUFFIXES

Different pieces of the directory tree can be stored in different databases, and then these databases can be distributed across multiple servers. The directory tree contains branch points called *nodes*. These nodes may be associated with databases. A suffix is a node of the directory tree associated with a particular database. For example, a simple directory tree might appear as illustrated in Figure 2.1, “A Directory Tree with One Root Suffix”.

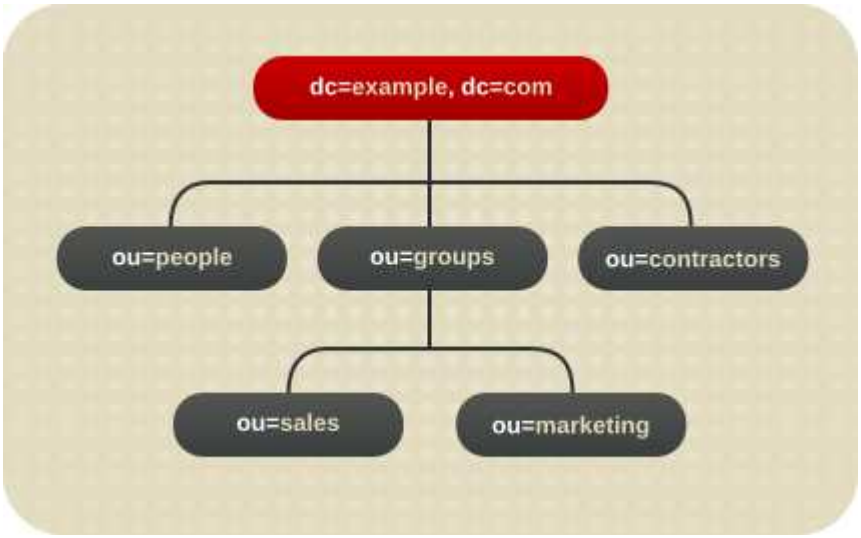


Figure 2.1. A Directory Tree with One Root Suffix

The **ou=people** suffix and all the entries and nodes below it might be stored in one database, the **ou=groups** suffix on another database, and the **ou=contractors** suffix on yet another database.

- Section 2.1.1, “Creating Suffixes”
- Section 2.1.2, “Maintaining Suffixes”

2.1.1. Creating Suffixes

Both root and sub suffixes can be created to organize the contents of the directory tree. A *root* suffix is the parent of a sub suffix. It can be part of a larger tree designed for the Directory Server. A *sub suffix* is a branch underneath a root suffix. The data for root and sub suffixes are contained by databases.

A directory might contain more than one root suffix. For example, an ISP might host several websites, one for **example.com** and one for **redhat.com**. The ISP would create two root suffixes, one corresponding to the **dc=example,dc=com** naming context and one corresponding to the **dc=redhat,dc=com** naming context, as shown in Figure 2.2, “A Directory Tree with Two Root Suffixes”.

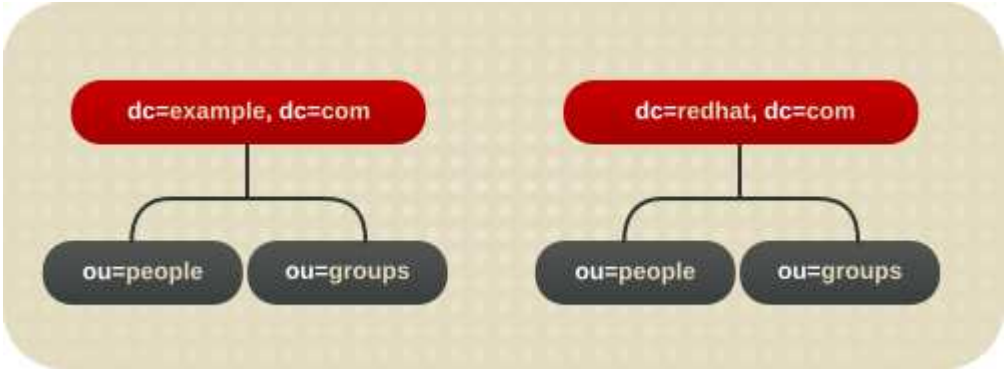


Figure 2.2. A Directory Tree with Two Root Suffixes

It is also possible to create root suffixes to exclude portions of the directory tree from search operations. For example, Example Corporation wants to exclude their European office from a search on the general Example Corporation directory. To do this, they create two root suffixes. One root suffix corresponds to the general Example Corporation directory tree, **dc=example,dc=com**, and one root suffix corresponds to the European branch of their directory tree, **l=europe,dc=example,dc=com**. From a client application's perspective, the directory tree looks as illustrated in Figure 2.3, "A Directory Tree with a Root Suffix Off Limits to Search Operations" .

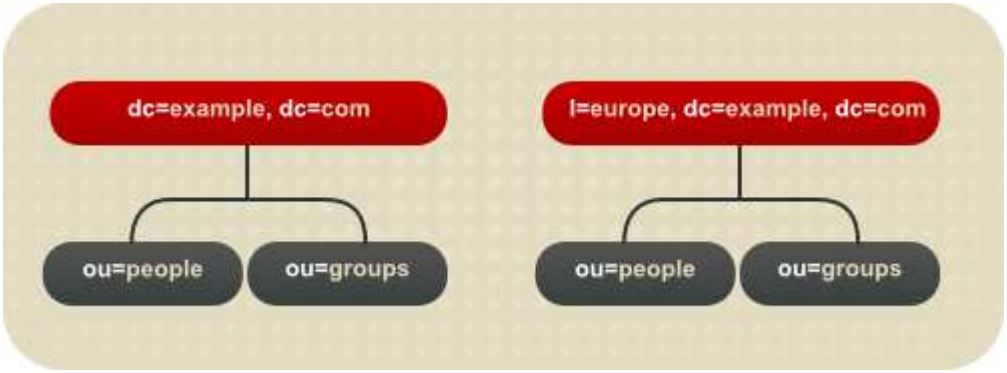


Figure 2.3. A Directory Tree with a Root Suffix Off Limits to Search Operations

Searches performed by client applications on the **dc=example,dc=com** branch of Example Corporation's directory will not return entries from the **l=europe,dc=example,dc=com** branch of the directory, as it is a separate root suffix.

If Example Corporation decides to include the entries in the European branch of their directory tree in general searches, they make the European branch a sub suffix of the general branch. To do this, they create a root suffix for Example Corporation, **dc=example,dc=com**, and then create a sub suffix beneath it for their European directory entries, **l=europe,dc=example,dc=com**. From a client application's perspective, the directory tree appears as illustrated in Figure 2.4, "A Directory Tree with a Sub Suffix" .

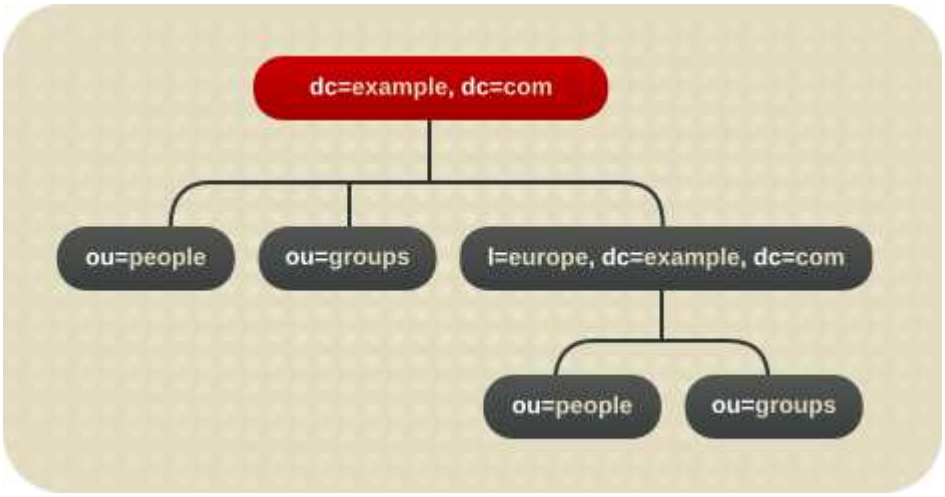


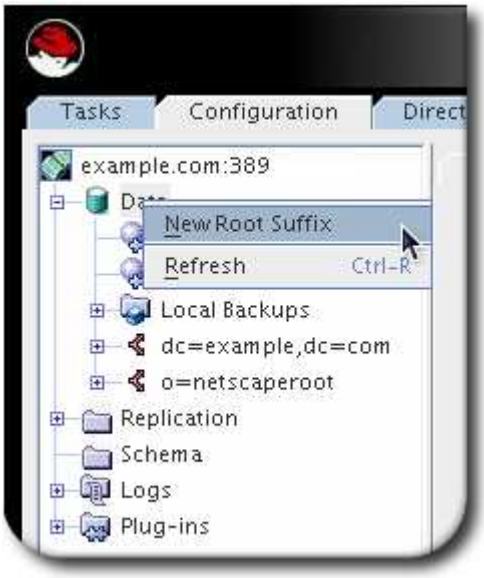
Figure 2.4. A Directory Tree with a Sub Suffix

This section describes creating root and sub suffixes for the directory using either the Directory Server Console or the command line.

- Section 2.1.1.1, "Creating a New Root Suffix Using the Console"
- Section 2.1.1.2, "Creating a New Sub Suffix Using the Console"
- Section 2.1.1.3, "Creating Root and Sub Suffixes from the Command Line"

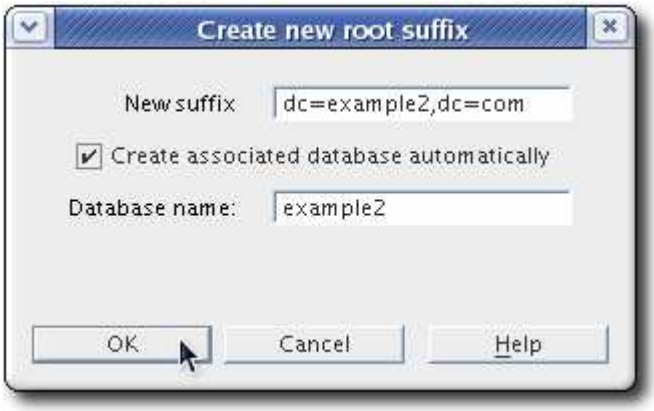
2.1.1.1. Creating a New Root Suffix Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Right-click **Data** in the left navigation pane, and select **New Root Suffix** from the pop-up menu.



3. Enter a unique suffix in the **New suffix** field.

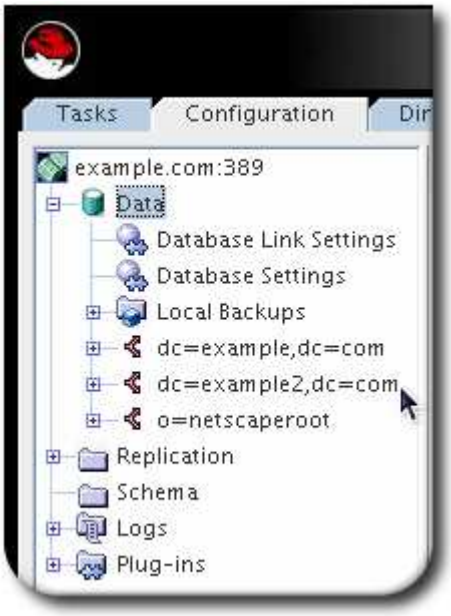
   The suffix must be named with **dc** naming conventions, such as **dc=example,dc=com**.



4. Select the **Create associated database automatically** to create a database at the same time as the new root suffix, and enter a unique name for the new database in the **Database name** field, such as **example2**. The name can be a combination of alphanumeric characters, dashes (**-**), and underscores (_). No other characters are allowed.

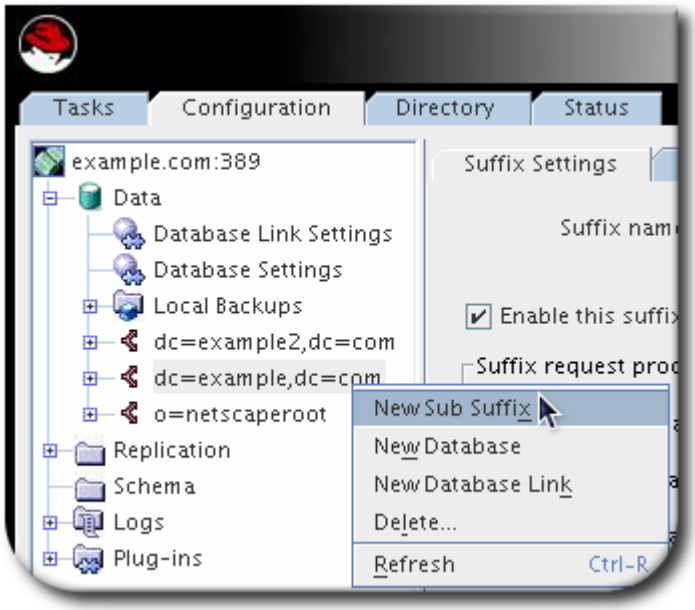
   Deselect the check box to create a database for the new root suffix later. This option specifies a directory where the database will be created. The new root suffix will be disabled until a database is created.

The new root suffix is listed under the **Data** folder.



2.1.1.2. Creating a New Sub Suffix Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Under the **Data** in the left navigation pane, select the suffix under which to add a new sub suffix. Right-click the suffix, and select **New Sub Suffix** from the pop-up menu.



   The **Create new sub suffix** dialog box is displayed.

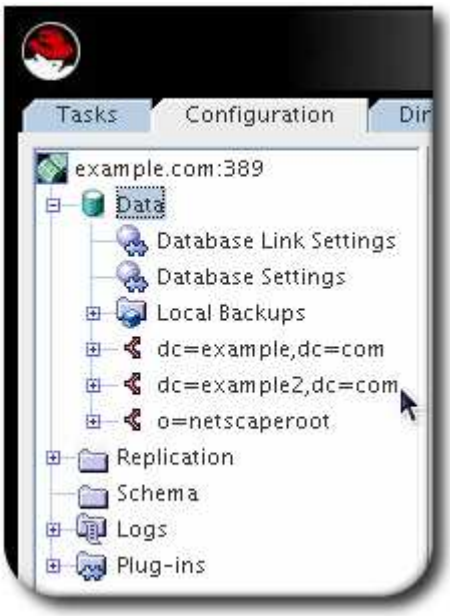
3. Enter a unique suffix name in the **New suffix** field. The suffix must be named in line with **dc** naming conventions, such as **ou=groups**.

The root suffix is automatically added to the name. For example, it the sub suffix **ou=groups** is created under the **dc=example,dc=com** suffix, the Console automatically names it **ou=groups,dc=example,dc=com**.

4. Select the **Create associated database automatically** check box to create a database at the same time as the new sub suffix, and enter a unique name for the new database in the **Database name** field, such as **example2**. The name can be a combination of alphanumeric characters, dashes (**-**), and underscores (_). No other characters are allowed.

   If the check box is not selected, than the database for the new sub suffix must be created later. The new sub suffix is disabled until a database is created.

The suffix appears automatically under its root suffix in the **Data** tree in the left navigation pane.



2.1.1.3. Creating Root and Sub Suffixes from the Command Line

Use the **ldapmodify** command-line utility to add new suffixes to the directory configuration file. The suffix configuration information is stored in the **cn=mapping tree,cn=config** entry.

NOTE

Avoid creating entries under the **cn=config** entry in the **dse.ldif** file. The **cn=config** entry in the simple, flat **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will suffer.

1. Add a new root suffix to the configuration file using the **ldapmodify** utility.

Example 2.1. Example Root Suffix Entry

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: nsMappingTree
nsslapd-state: backend
nsslapd-backend: UserData
cn: dc=example,dc=com
```

2. Create a sub suffix for groups under this root suffix using **ldapmodify** to add the sub suffix entry:

```
dn: cn=ou=groups\,dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: nsMappingTree
nsslapd-state: backend
nsslapd-backend: GroupData
nsslapd-parent-suffix: dc=example,dc=com
cn: ou=groups,dc=example,dc=com
```

NOTE

To maintain suffixes using the Directory Server Console, respect the same spacing used to name the root and sub suffixes in the command line. For example, if a root suffix is named **ou=groups ,dc=example,dc=com**, with two spaces after **groups**, any sub suffixes created under this root will need to specify two spaces after **ou=groups**, as well.

The following table describes the attributes used to configure a suffix entry:

Table 2.1. Suffix Attributes

Attribute Name	Value
dn	Defines the DN for the suffix. The DN is contained in quotes. The value entered takes the form **cn="dc=example,dc=com",cn=mapping tree,cn=config**. This attribute is required.
cn	Defines the relative DN (RDN) of the entry. This attribute is required.
objectclass	Tells the server that the entry is root or sub suffix entry. It always takes the value **nsMappingTree**. This attribute is required.
nsslapd-state	Determines how the suffix handles operations. This attribute takes the following values: • **backend**: The back end (database) is used to process all operations. • **disabled**: The database is not available for processing operations. The server returns a *No such search object* error in response to requests made by client applications. • **referral**: A referral is returned for requests made to this suffix. • **referral on update**: The database is used for all operations except update requests, which receive a referral. The default value is **disabled**.

Attribute Name	Value
nsslapd-referral	Defines the LDAP URL of the referral to be returned by the suffix. This attribute can be multi-valued, with one referral per value. This attribute is required when the value of the ***nsslapd-state*** attribute is **referral** or **referral on update**.
nsslapd-backend	Gives the name of the database or database link used to process requests. This attribute can be multi-valued, with one database or database link per value. See Section 2.3, "Creating and Maintaining Database Links" for more information about database links. This attribute is required when the value of the ***nsslapd-state*** attribute is set to **backend** or **referral on update**.
nsslapd-distribution-plugin	Specifies the shared library to be used with the custom distribution function. This attribute is required only when more than one database is specified in the ***nsslapd-backend*** attribute. See Section 2.2, "Creating and Maintaining Databases" for more information about the custom distribution function.
nsslapd-distribution-funct	Specifies the name of the custom distribution function. This attribute is required only when more than one database is specified in the ***nsslapd-backend*** attribute. See Section 2.2, "Creating and Maintaining Databases" for more information about the custom distribution function.
nsslapd-parent-suffix	Provides the DN of the parent entry for a sub suffix. By default, this attribute is not present, which means that the suffix is regarded as a root suffix. For example, to create a sub suffix names **o=sales,dc=example,dc=com** under the root suffix **dc=example,dc=com**, add **nsslapd-parent-suffix: dc=example,dc=com** to the sub suffix.

2.1.2. Maintaining Suffixes

- Section 2.1.2.1, "Viewing the Default Naming Context"
- Section 2.1.2.2, "Disabling a Suffix"
- Section 2.1.2.3, "Deleting a Suffix"

2.1.2.1. Viewing the Default Naming Context

A naming context is analogous to the suffix; it is the root structure for naming directory entries. There can be multiple naming contexts, depending on the directory and data structure; for example, a standard Directory Server configuration has a user suffix such as **dc=example,dc=com**, a configuration suffix in **cn=config**, and an administrative configuration suffix in **o=netscaperoot**.

Many directory trees have multiple naming contexts to be used with different types of entries or with logical data divisions. Clients which access the Directory Server may not know what naming context they need to use. The Directory Server has a server configuration attribute which signals to clients what the default naming context is, if they have no other naming context configuration known to them.

The default naming context is set in the ***nsslapd-defaultnamingcontext*** attribute in **cn=config**. This value is propagated over to the root DSE and can be queried by clients anonymously by checking the ***defaultnamingcontext*** attribute in the root DSE.

For example:

```
[root@server ~]# ldapsearch -p 389 -h server.example.com -x -b "" -s base | egrep namingcontext
namingContexts: dc=example,dc=com
namingContexts: dc=example,dc=net
namingContexts: dc=redhat,dc=com
defaultnamingcontext: dc=example,dc=com
```

IMPORTANT

By default, the ***nsslapd-defaultnamingcontext*** attribute is included in the list of attributes which *can* be deleted, in the ***nsslapd-allowed-to-delete-attrs*** attribute. This allows the current default suffix to be deleted and then updates the server configuration accordingly.

If for some reason the ***nsslapd-defaultnamingcontext*** attribute is removed from the list of configuration attributes which can be deleted, then no changes to that attribute are preserved. If the default suffix is deleted, that change cannot be propagated to the server configuration. This means that the ***nsslapd-defaultnamingcontext*** attribute retains the old information instead of being blank (removed), which is the correct and current configuration.

To maintain configuration consistency, do not remove the ***nsslapd-defaultnamingcontext*** attribute from the ***nsslapd-allowed-to-delete-attrs*** list.

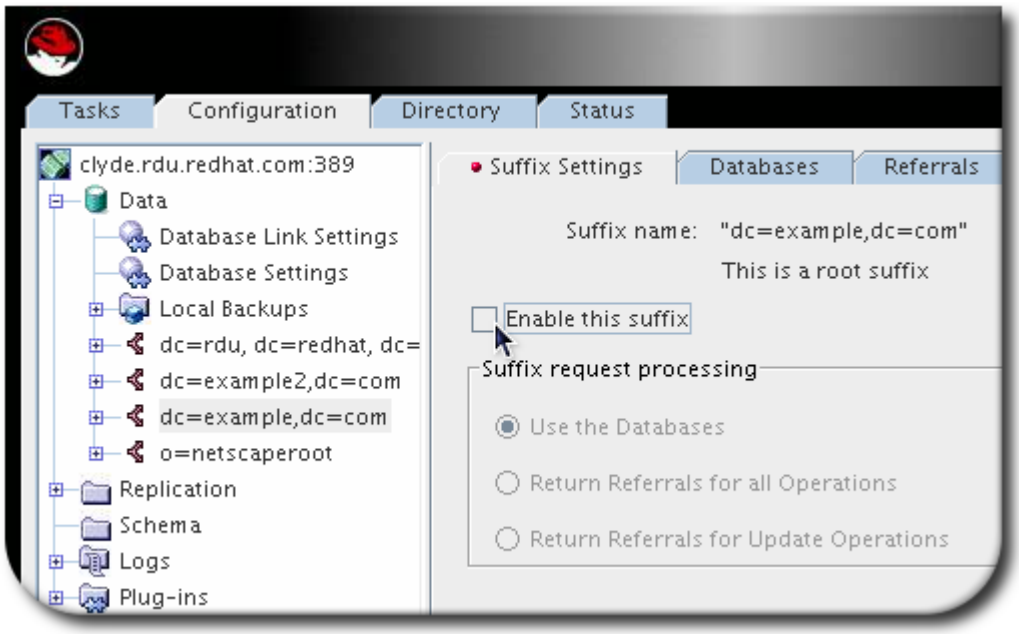
2.1.2.2. Disabling a Suffix

Sometimes, a database may need taken down for maintenance, but the data the database contains are not replicated. Rather than returning a referral, disable the suffix responsible for the database.

Once a suffix is disabled, the contents of the database related to the suffix are invisible to client applications when they perform LDAP operations such as search, add, and modify.

To disable a suffix:

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left navigation pane, click the suffix to disable.
3. Click the **Suffix Setting** tab, and deselect the **Enable this suffix** check box.



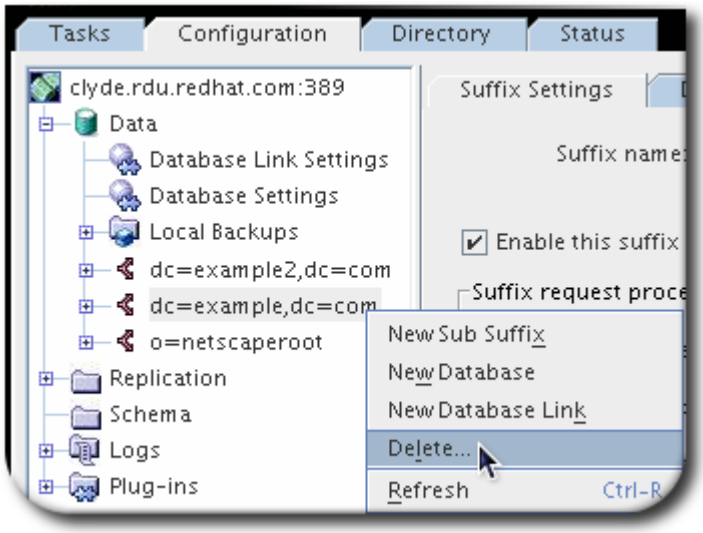
2.1.2.3. Deleting a Suffix

WARNING

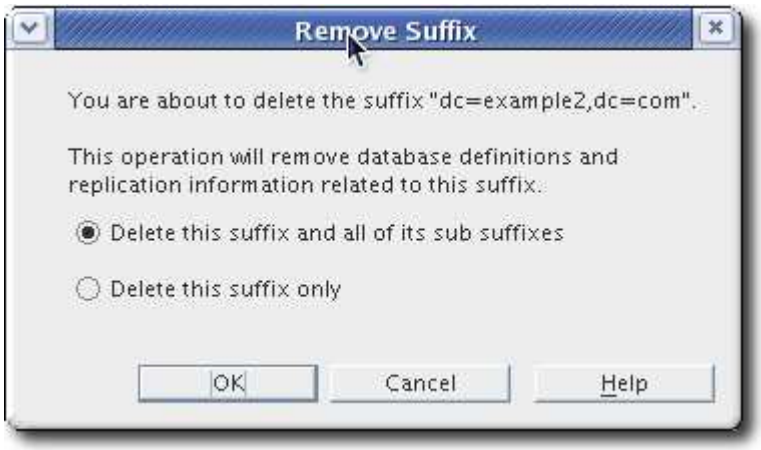
Deleting a suffix also deletes all database entries and replication information associated with that suffix.

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left navigation pane, select the suffix to delete.

3. Right-click the suffix, and select **Delete** from the menu.



4. Select either **Delete this suffix and all of its sub suffixes** or **Delete this suffix only**.



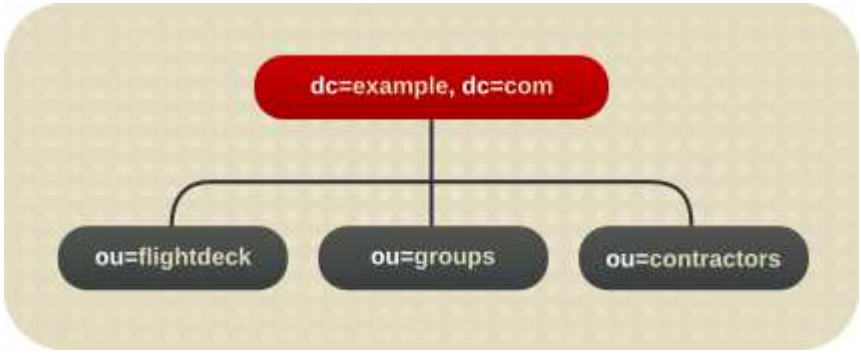
2.2. CREATING AND MAINTAINING DATABASES

After creating suffixes to organizing the directory data, create databases to contain that directory data. Databases are used to store directory data.

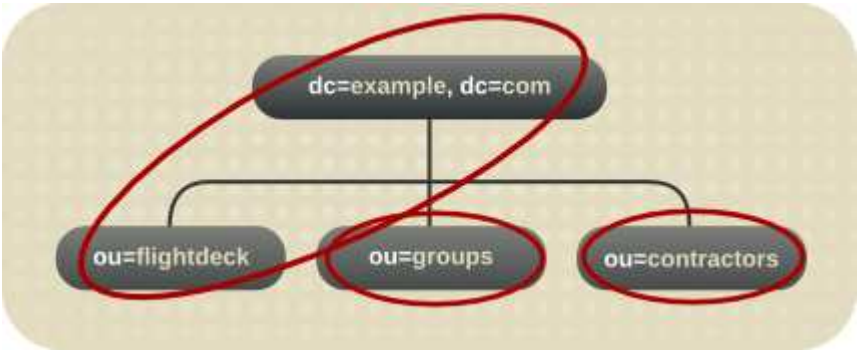
2.2.1. Creating Databases

The directory tree can be distributed over multiple Directory Server databases. There are two ways to distribute data across multiple databases:

- One database per suffix. The data for each suffix is contained in a separate database.



Three databases are added to store the data contained in separate suffixes.

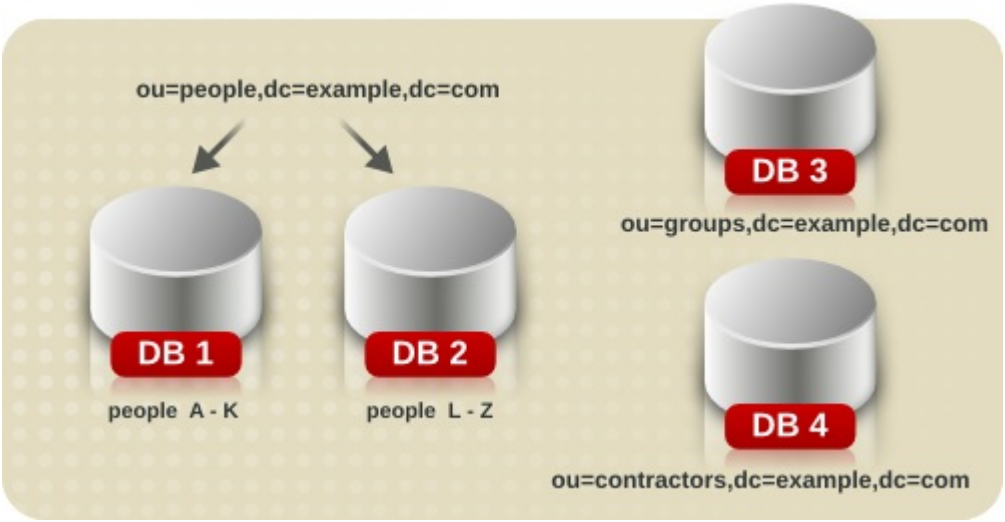


This division of the tree corresponds to three databases.

Database one contains the data for **ou=people** plus the data for **dc=example,dc=com**, so that clients can conduct searches based at **dc=example,dc=com**. Database two contains the data for **ou=groups**, and database three contains the data for **ou=contractors**.

- Multiple databases for one suffix.

  Suppose the number of entries in the **ou=people** branch of the directory tree is so large that two databases are needed to store them. In this case, the data contained by **ou=people** could be distributed across two databases.



DB1 contains people with names from A-K, and **DB2** contains people with names from L-Z. **DB3** contains the **ou=groups** data, and **DB4** contains the **ou=contractors** data.

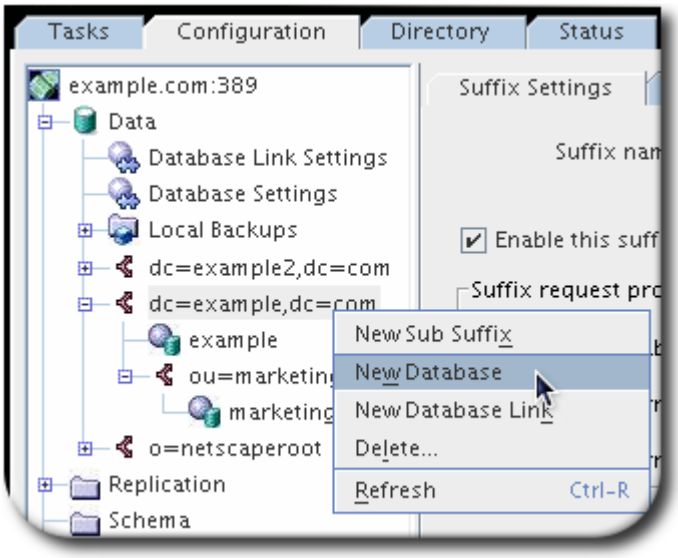
Custom distribution plug-in distributes data from a single suffix across multiple databases. Contact Red Hat Professional Services for information on how to create distribution logic for Directory Server.

2.2.1.1. Creating a New Database for an Existing Suffix Using the Console

1. In the Directory Server Console, select the **Configuration** tab.

2. In the left pane, expand **Data**, then click the suffix to which to add the new database.

3. Right-click the suffix, and select **New Database** from the pop-up menu.



4. Enter a unique name for the database, such as **example2**. The database name can be a combination of alphanumeric characters, dashes (**-**), and underscores (_).

The **Create database in** field is automatically filled with the default database directory (**/var/lib/dirsrv/slapd-*instance_name*/db**) and the name of the new database. It is also possible to enter or browse for a different directory location.

2.2.1.2. Creating a New Database for a Single Suffix from the Command Line

Use the **ldapmodify** command-line utility to add a new database to the directory configuration file. The database configuration information is stored in the **cn=ldbm database,cn=plugins,cn=config** entry.

For example, add a new database to the server **example1**:

1. Run **ldapmodify** and create the entry for the new database.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=UserData,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: ou=people,dc=example,dc=com
```

The entry added corresponds to a database named **UserData** that contains the data for the root or sub suffix **ou=people,dc=example,dc=com**.

2. Create a root or sub suffix, as described in Section 2.1.1.3, "Creating Root and Sub Suffixes from the Command Line". The database name, given in the DN attribute, must correspond with the value in the ***nsslapd-backend*** attribute of the suffix entry.

2.2.1.3. Adding Multiple Databases for a Single Suffix

A single suffix can be distributed across multiple databases. However, to distribute the suffix, a custom distribution function has to be created to extend the directory. For more information on creating a custom distribution function, contact Red Hat Professional Services.

NOTE

Once entries have been distributed, they cannot be redistributed. The following restrictions apply:

- The distribution function cannot be changed once entry distribution has been deployed.
- The LDAP **modrdn** operation cannot be used to rename entries if that would cause them to be distributed into a different database.
- Distributed local databases cannot be replicated.
- The **ldapmodify** operation cannot be used to change entries if that would cause them to be distributed into a different database.

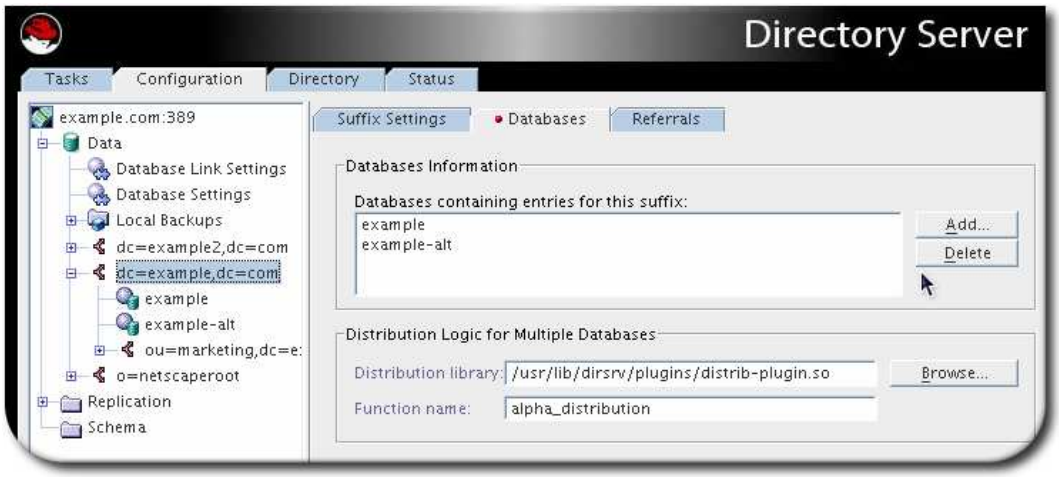
Violating these restrictions prevents Directory Server from correctly locating and returning entries.

After creating a custom distribution logic plug-in, add it to the directory.

The distribution logic is a function declared in a suffix. This function is called for every operation reaching this suffix, including subtree search operations that start above the suffix. A distribution function can be inserted into a suffix using both the Console and the command line.

2.2.1.3.1. Adding the Custom Distribution Function to a Suffix Using the Directory Server Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand **Data** in the left navigation pane. Select the suffix to which to apply the distribution function.
3. Select the **Databases** tab in the right window.



4. The databases associated with the suffix are already listed in the **Databases** tab. Click **Add** to associate additional databases with the suffix.
5. Enter the path to the distribution library.
6. Enter the name of the distribution function in the **Function name** field.

2.2.1.3.2. Adding the Custom Distribution Function to a Suffix Using the Command Line

1. Run **ldapmodify**.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Add the following attributes to the suffix entry itself, supplying the information about the custom distribution logic:

```
dn: suffix
changetype: modify
add: nsslapd-backend
nsslapd-backend: Database1
-
add: nsslapd-backend
nsslapd-backend: Database2
-
add: nsslapd-backend
nsslapd-backend: Database3
-
add: nsslapd-distribution-plugin
nsslapd-distribution-plugin: /full/name/of/a/shared/library
-
add: nsslapd-distribution-funct
nsslapd-distribution-funct: distribution-function-name
```

The ***nsslapd-backend*** attribute specifies all of the databases associated with this suffix. The ***nsslapd-distribution-plugin*** attribute specifies the name of the library that the plug-in uses. The ***nsslapd-distribution-funct*** attribute provides the name of the distribution function itself.

For more information about using the **ldapmodify** command-line utility, see Section 3.2.4, "Adding and Modifying Entries Using ldapmodify".

2.2.2. Maintaining Directory Databases

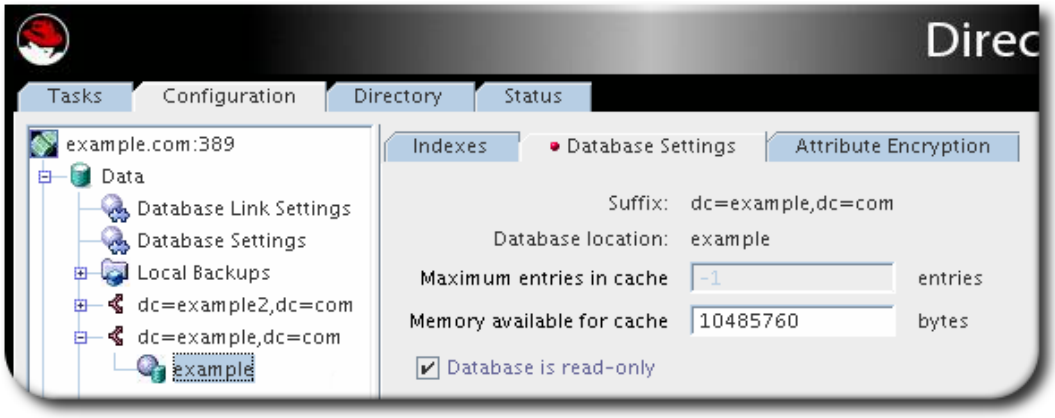
2.2.2.1. Placing a Database in Read-Only Mode

When a database is in read-only mode, you cannot create, modify, or delete any entries. One of the situations when read-only mode is useful is for manually initializing a consumer or before backing up or exporting data from the Directory Server. Read-only mode ensures a faithful image of the state of these databases at a given time.

The Directory Server Console and the command-line utilities do not automatically put the directory in read-only mode before export or backup operations because this would make your directory unavailable for updates. However, with multi-master replication, this might not be a problem.

2.2.2.1.1. Making a Database Read-Only Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand **Data** in the left pane. Expand the suffix containing the database to put in read-only mode.
3. Select the database to put into read-only mode.
4. Select the **Database Settings** tab in the right pane.



5. Select the **database is read-only** check box.

The change takes effect immediately.

Before importing or restoring the database, ensure that the databases affected by the operation are *not* in read-only mode.

To disable read-only mode, open the database up in the Directory Server Console again and uncheck the **database is read-only** check box.

2.2.2.1.2. Making a Database Read-Only from the Command Line

To manually place a database into read-only mode:

1. Run **ldapmodify**.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Change the read-only attribute to **on**

```
dn: cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-readonly
nsslapd-readonly: on
```

NOTE

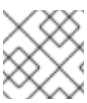
By default, the name of the database created at installation time is **userRoot**.

2.2.2.1.3. Placing the Entire Directory Server in Read-Only Mode

If the Directory Server maintains more than one database and all databases need to be placed in read-only mode, this can be done in a single operation.

WARNING

This operation also makes the Directory Server configuration read-only; therefore, you cannot update the server configuration, enable or disable plug-ins, or even restart the Directory Server while it is in read-only mode. Once read-only mode is enabled, it *cannot* cannot be undone from the Console; you must modify the configuration files.

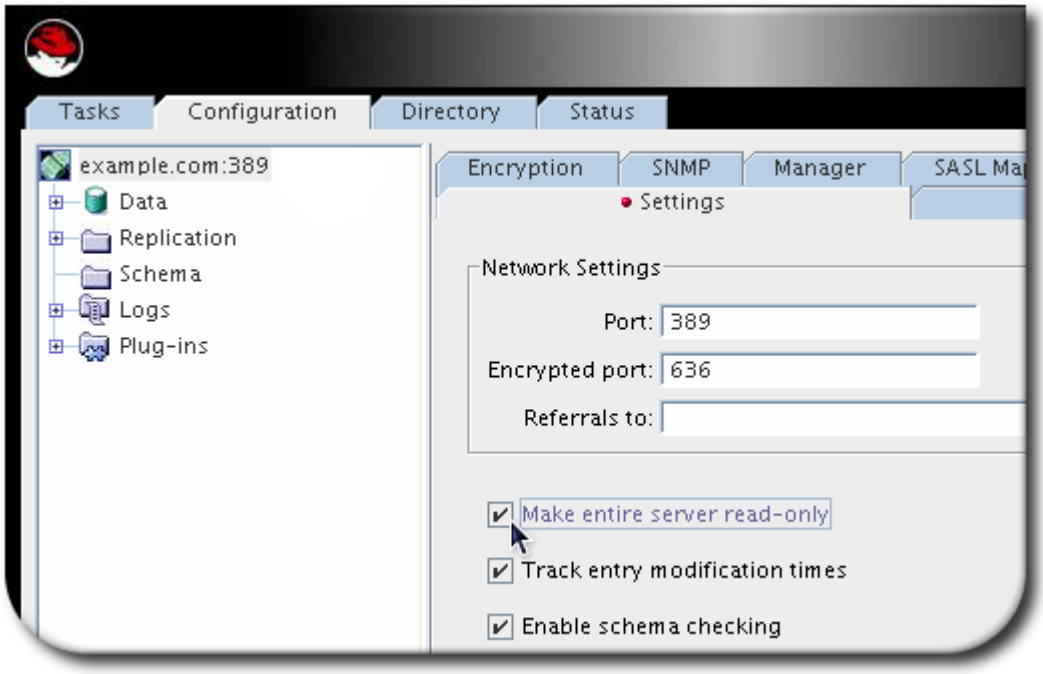


NOTE

If Directory Server contains replicas, *do not* use read-only mode because it will disable replication.

To put the Directory Server in read-only mode:

1. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
2. Select the **Settings** tab in the right pane.

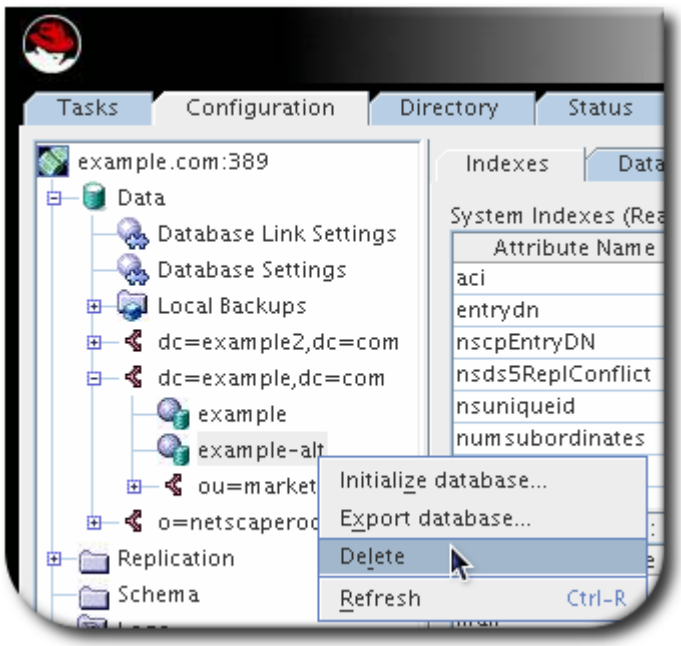


3. Select the **Make Entire Server Read-Only** check box.
4. Click **Save**, and then restart the server.

2.2.2.2. Deleting a Database

Deleting a database deletes the configuration information and entries for that database only, not the physical database itself.

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder, and then select the suffix.
3. Select the database to delete.
4. Right-click the database and select **Delete** from the pop-up menu.



5. Confirm that the database should be deleted in the **Delete Database** dialog box.

2.2.2.3. Configuring Transaction Logs for Frequent Database Updates

When the server is going to be asked to perform frequent database updates (LDAP adds, modifies, replication), the database transaction log files should be configured to be on a different disk than the primary database files.

Storing the transaction log files on a separate physical disk improves performance because the disk heads do not thrash moving between the log files and the data files.

1. Stop the Directory Server instance.

```
service dirsrv stop example
```

2. Create the new directory, if necessary, where the transaction logs will be located.

```
mkdir /home/exampledb-txnlogs
```

3. Set the appropriate file permissions on the directory so that the Directory Server user can access it; the default Directory Server user and group is **nobody:nobody**. However, Red Hat strongly recommends to use a different user and group name such as **dirsrv** during the installation.

```
chown nobody:nobody /home/exampledb-txnlogs
```

4. Open the **dse.ldif** file in the Directory Server instance's configuration directory.

```
[root@server ~]# vim /etc/dirsrv/slapd-instance_name/dse.ldif
```

5. Change the **nsslapd-db-logdirectory** directive for the new log file path:

```
nsslapd-db-logdirectory: /home/exampledb-txnlogs
```

   This attribute goes on the same entry that has the **nsslapd-dbcachesize** attribute.

6. Open the database directory.

```
[root@server ~]# cd /var/lib/dirsrv/slapd-instance_name/db
```

7. Remove all of the **__db.*** files.

8. Move the **log.*** files to the new location.

9. Start the Directory Server instance again.

```
[root@server ~]# service dirsrv start example
```

2.2.3. Configuring Attribute Encryption

The Directory Server offers a number of mechanisms to secure access to sensitive data, such as access control rules to prevent unauthorized users from reading certain entries or attributes within entries and SSL to protect data from eavesdropping and tampering on untrusted networks. However, if a copy of the server's database files should fall into the hands of an unauthorized person, they could potentially extract sensitive information from those files. Because information in a database is stored in plain text, some sensitive information, such as government identification numbers or passwords, may not be protected enough by standard access control measures.

For highly sensitive information, this potential for information loss could present a significant security risk. In order to remove that security risk, Directory Server allows portions of its database to be encrypted. Once encrypted, the data are safe even in the event that an attacker has a copy of the server's database files.

Database encryption allows attributes to be encrypted in the database. Both encryption and the encryption cipher are configurable per attribute per back end. When configured, every instance of a particular attribute, even index data, is encrypted for every entry stored in that database.

NOTE

There is one exception to encrypted data: any value which is used as the RDN for an entry is not encrypted within the entry DN. For example, if the ***uid*** attribute is encrypted, the value is encrypted in the entry but is displayed in the DN:

```
# entry-id: 16
dn: uid=jsmith1234,ou=People,dc=example,dc=com
nsUniqueId: ee91ea82-1dd111b2-9f36e9bc-39fb8550
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
givenName: John
sn: Smith
uid:: Sf04P9nJWGU1qiW9JJCGRg==
```

That would allow someone to discover the encrypted value.

Any attribute used within the entry DN cannot be effectively encrypted, since it will always be displayed in the DN. Be aware of what attributes are used to build the DN and design the attribute encryption model accordingly.

Indexed attributes may be encrypted, and attribute encryption is fully compatible with indexing. The contents of the index files that are normally derived from attribute values are also encrypted to prevent an attacker from recovering part or all of the encrypted data from an analysis of the indexes.

Since the server pre-encrypts all index keys before looking up an index for an encrypted attribute, there is some effect on server performance for searches that make use of an encrypted index, but the effect is not serious enough that it is no longer worthwhile to use an index.

2.2.3.1. Encryption Keys

In order to use attribute encryption, the server must be configured for SSL and have SSL enabled because attribute encryption uses the server's SSL encryption key and the same PIN input methods as SSL. The PIN must either be entered manually upon server startup or a PIN file must be used.

Randomly generated symmetric cipher keys are used to encrypt and decrypt attribute data. A separate key is used for each configured cipher. These keys are *wrapped* using the public key from the server's SSL certificate, and the resulting wrapped key is stored within the server's configuration files. The effective strength of the attribute encryption is never higher than the strength of the server's SSL key used for wrapping. Without access to the server's private key, it is not possible to recover the symmetric keys from the wrapped copies.

WARNING

There is no mechanism for recovering a lost key. Therefore, it is especially important to back up the server's certificate database safely. If the server's certificate were lost, it would not be possible to decrypt any encrypted data stored in its database.

WARNING

If the SSL certificate is expiring and needs to be renewed, export the encrypted back end instance before the renewal. Update the certificate, then re-import the exported LDIF file.

2.2.3.2. Encryption Ciphers

The encryption cipher is configurable on a per-attribute basis and must be selected by the administrator at the time encryption is enabled for an attribute. Configuration can be done through the Console or through the command line.

The following ciphers are supported:

- Advanced Encryption Standard (AES)
- Triple Data Encryption Standard (3DES)

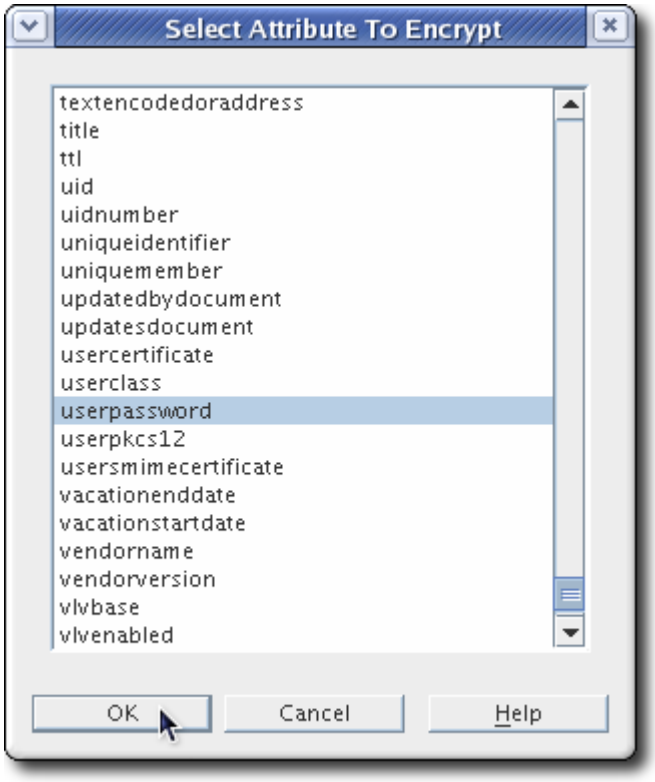
All ciphers are used in Cipher Block Chaining mode.

Once the encryption cipher is set, it should not be changed without exporting and re-importing the data.

2.2.3.3. Configuring Attribute Encryption from the Console

1. In the **Configuration** tab, select the **Data** node.
2. Expand the suffix, and select the database to edit.
3. Select the **Attribute Encryption** tab.

4. Click the **Add Attribute** button to open the list of attributes. Select the attribute to encrypt.



NOTE

For existing attribute values to be encrypted, the information must be exported from the database, then re-imported. See Section 2.2.3.6, "Exporting and Importing an Encrypted Database".

5. Select which encryption cipher to use.

NOTE

The encryption cipher to use is set separately for each attribute, so attribute encryption is applied to each attribute one at a time.

To remove encryption from attributes, select them from the list of encrypted attributes in the **Attribute Encryption** table, click the **Delete** button, and then click **Save** to apply the changes. Any deleted attributes have to be manually re-added after saving.

2.2.3.4. Configuring Attribute Encryption Using the Command Line

1. Run the **ldapmodify** command:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Add an encryption entry for the attribute being encrypted. For example, this entry encrypts the ***telephoneNumber*** attribute with the AES cipher:

```
dn: cn=telephoneNumber,cn=encrypted attributes,cn=Database1,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectclass: top
objectclass: nsAttributeEncryption
cn: telephoneNumber
nsEncryptionAlgorithm: AES
```

3. For existing attributes in entries to be encrypted, the information must be exported, then re-imported. See Section 2.2.3.6, “Exporting and Importing an Encrypted Database” .

For more information on attribute encryption configuration schema, see "Database Attributes under cn=attributeName,cn=encrypted attributes,cn=database_name,cn=ldbm database,cn=plugins,cn=config" in the *Directory Server Configuration and Command-Line Tool Reference* .

2.2.3.5. Enabling Attribute Encryption for Existing Attribute Values

To enable attribute encryption on an attribute with existing stored data, export the database to LDIF first, then make the configuration change, then re-import the data to the database. The server does not enforce consistency between encryption configuration and stored data; therefore, pay careful attention that all existing data are exported before enabling or disabling encryption.

2.2.3.6. Exporting and Importing an Encrypted Database

Exporting and importing encrypted databases is similar to exporting and importing regular databases. However, the encrypted information must be decrypted when it is exported to LDIF, then re-encrypted when it is imported to the database. Using the **-E** option when running the **db2ldif** and **ldif2db** scripts will decrypt the data on export and re-encrypt it on import.

1. Export the data using the **db2ldif** script, as follows:

```
db2ldif -n Database1 -E -a /path/to/output.ldif -s "dc=example,dc=com" -s "o=userRoot"
```

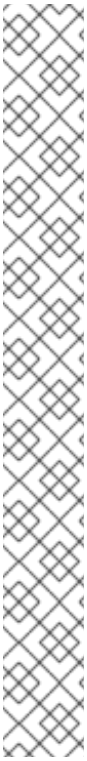
See Section 4.2.3, “Exporting to LDIF from the Command Line” for more information.

2. Make any configuration changes.

3. Re-import the data using the **ldif2db** script, as follows:

```
ldif2db -n Database1 -E -i /path/to/output.ldif
```

See Section 4.1.6, "Importing from the Command Line" for more information.



NOTE

When enabling encryption for data that is already present in the database, several additional security concerns arise:

- It is possible for old, unencrypted data to persist in the server's database page pool backing file, even after a successful re-import with encryption. To remove this data, stop the server and delete the **db/guardian** file, then re-start the server. This will force recovery, a side-effect of which is deleting the backing file. However, it is possible that the data from the deleted file could still be recovered from the hard drive unless steps are taken to overwrite the disk blocks that it occupied.
- After enabling encryption and importing data, be sure to delete the LDIF file because it contains plain text values for the now-encrypted data. Ensure that the disk blocks that it occupied are overwritten.
- The unencrypted data previously stored in the server's database may persist on disk after a successful re-import with encryption. This is because the old database files are deleted as part of the import process. Ensure that the disk blocks that those files occupied are overwritten.
- Data stored in the server's replication log database is never encrypted; therefore, care should be taken to protect those files if replication is used.
- The server does not attempt to protect unencrypted data stored in memory. This data may be copied into a system page file by the operating system. For this reason, ensure that any page or swap files are adequately protected.

2.2.3.7. Updating Attribute Encryption Keys for New SSL/TLS Certificates

When SSL is first configured, there is no problem with attribute encryption. However, if the SSL certificate is changed, then attribute encryption fails, with messages like these:

```
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - attrcrypt_unwrap_key: failed to unwrap key for
cipher AES
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to retrieve key for cipher AES in
attrcrypt_cipher_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to initialize cipher AES in attrcrypt_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - attrcrypt_unwrap_key: failed to unwrap key for
cipher AES
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to retrieve key for cipher AES in
attrcrypt_cipher_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to initialize cipher AES in attrcrypt_init
```

This is because the previously-generated keys do not work with the new server certificate. To correct these errors, force the server to generate new keys for attribute encryption:

1. Stop the server.

```
service dirsrv stop
```

2. Open the **dse.ldif** file.

```
vim /etc/dirsrv/dse.ldif
```

3. There are special encryption key entries for the encryption ciphers used for attribute encryption under the database configuration. For example:

```
dn: cn=AES,cn=encrypted attribute keys,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: extensibleObject
```

```
cn: AES
nssymmetrickey:: mSLm/RlCLvPZrSdARHPowedF9zKx+kjVTww5ARE4w0lbl2YlYvrl3bNg=
```

Delete these entries.

4. Start the server again.

```
service dirsrv start
```

2.3. CREATING AND MAINTAINING DATABASE LINKS

Chaining means that a server contacts other servers on behalf of a client application and then returns the combined results. Chaining is implemented through a *database link*, which points to data stored remotely. When a client application requests data from a database link, the database link retrieves the data from the remote database and returns it to the client.

- Section 2.3.1, "Creating a New Database Link"
- Section 2.3.2, "Configuring the Chaining Policy"
- Section 2.3.3, "Maintaining Database Links"
- Section 2.3.4, "Configuring Database Link Defaults"
- Section 2.3.5, "Deleting Database Links"
- Section 2.3.6, "Database Links and Access Control Evaluation"

For more general information about chaining, see the chapter "Designing the Directory Topology," in the *Directory Server Deployment Guide*. Section 15.9, "Monitoring Database Link Activity" covers how to monitor database link activity.

2.3.1. Creating a New Database Link

The basic database link configuration requires four piece of information:

- *Suffix information.* A suffix is created in the directory tree that is managed by the database link, not a regular database. This suffix corresponds to the suffix on the remote server that contains the data.
- *Bind credentials.* When the database link binds to a remote server, it impersonates a user, and this specifies the DN and the credentials for each database link to use to bind with remote servers.
- *LDAP URL.* This supplies the LDAP URL of the remote server to which the database link connects. The URL consists of the protocol (ldap or ldaps), the host name or IP address (IPv4 or IPv6) for the server, and the port.
- *List of failover servers.* This supplies a list of alternative servers for the database link to contact in the event of a failure. This configuration item is optional.

NOTE

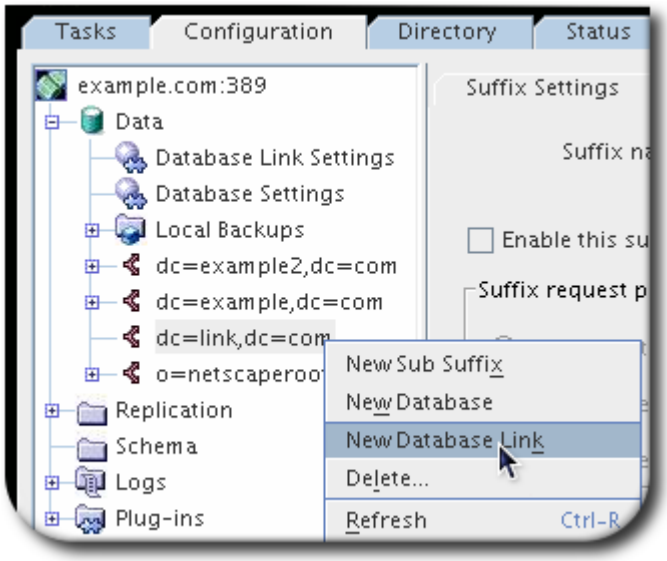
If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any chaining operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

2.3.1.1. Creating a New Database Link Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Create a new suffix as described in Section 2.1.1, "Creating Suffixes".

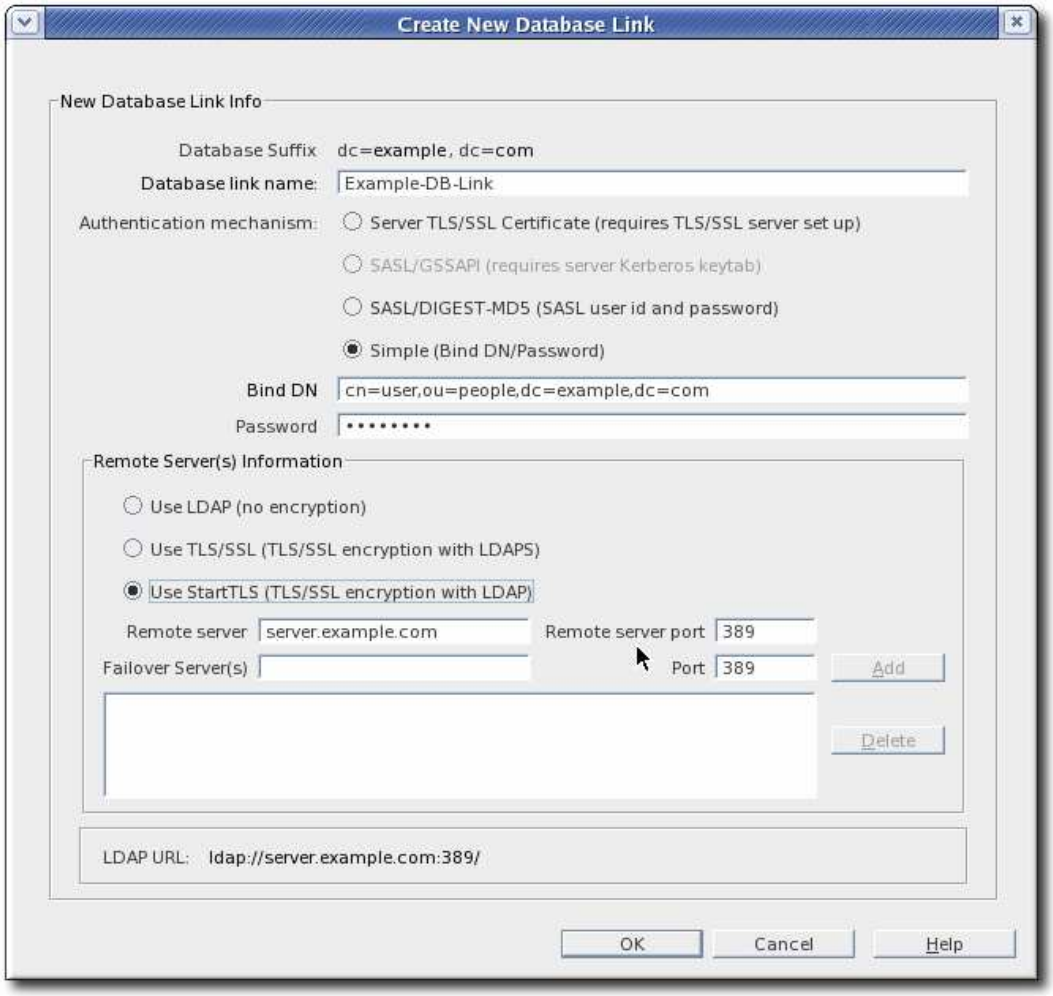
   Deselect the **Create associated database automatically** check box. It is simpler to configure a database link on a suffix without a database associated with it because having both a database and database link requires custom distribution functions to distribute directory data.

3. In the left pane, right-click the new suffix, and select **New Database Link** from the pop-up menu.



4. Fill in the database link name. The name can be a combination of alphanumeric characters, dashes (**-**), and underscores (_). No other characters, like spaces, are allowed.

5. Set the radio button for the appropriate method for authentication.



There are four authentication methods:

- *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the user as whom the server connects to the remote server.
- *Server TLS/SSL Certificate* uses the local server's SSL certificate to authenticate to the remote server. A certificate must be installed on the local server for certificate-based authentication, and the remote server must have certificate mapping configured so that it can map the subject DN in the local server's certificate to the corresponding user entry.

  Configuring SSL and certificate mapping is described in Section 7.4, "Setting up TLS/SSL".

  NOTE

  When the database link and remote server are configured to communicate using SSL, this does not mean that the client application making the operation request must also communicate using SSL. The client can bind using a normal port.

- *SASL/DIGEST-MD5* requires only the bind DN and password to authenticate.
- *SASL/GSSAPI* requires the local server to have a Kerberos keytab (as in Section 7.12.2.2, "About the KDC Server and Keytabs"), and the remote server to have a SASL mapping to map the local server's principal to the real user entry (as in Section 7.11.3.1, "Configuring SASL Identity Mapping from the Console").

6. In the **Remote Server Information** section, select the connection type for the local server to use to connect to the remote server. There are three options:
   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS/SSL.

     When using SSL/TLS, make sure that the remote server's port number is set to its secure port.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any chaining operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

7. In the **Remote Server Information** section, fill in the name (host name, IPv4 address, or IPv6 address) and port number for the remote server.

   For any failover servers, fill in the host name and port number, and click the **Add** button. A failover server is a backup server, so that if the primary remote server fails, the database link contacts the first server in the failover servers list and cycles through the list until a server is accessed.

The new database link is listed under the suffix, in place of the database.

NOTE

The Console provides a checklist of information that needs to be present on the *remote* server for the database link to bind successfully. To view this checklist, click the new database link, and click the **Authentication** tab. The checklist is in the **Remote server checklist** box.

2.3.1.2. Creating a Database Link from the Command Line

1. Use the **ldapmodify** command-line utility to create a new database link. The new instance must be located in the **cn=chaining database,cn=plugins,cn=config** entry.

   ```
   ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
   ```

2. Specify the configuration information for the database link:

   ```
   dn: cn=examplelink,cn=chaining database,cn=plugins,cn=config
   changetype: add
   objectclass: top
   objectclass: extensibleObject
   objectclass: nsBackendInstance
   nsslapd-suffix: ou=people,dc=example,dc=com suffix being chained
   nsfarmserverurl: ldap://people.example.com:389/  LDAP URL to remote server
   nsMultiplexorBindDN: cn=proxy admin,cn=config bind DN
   nsMultiplexorCredentials: secret bind password
   cn: examplelink
   ```

NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any chaining operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

Default configuration attributes are contained in the **cn=default instance config,cn=chaining database,cn=plugins,cn=config** entry. These configuration attributes apply to all database links at creation time. Changes to the default configuration only affect new database links. The default configuration attributes on existing database links cannot be changed.

Each database link contains its own specific configuration information, which is stored with the database link entry itself, **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config**. For more information about configuration attributes, see the *Directory Server Configuration and Command-Line Tool Reference* .

- Section 2.3.1.2.1, "Providing Suffix Information"
- Section 2.3.1.2.2, "Providing Bind Credentials"

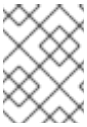
- Section 2.3.1.2.3, "Providing an LDAP URL"
- Section 2.3.1.2.4, "Providing a List of Failover Servers"
- Section 2.3.1.2.5, "Using Different Bind Mechanisms"
- Section 2.3.1.2.6, "Summary of Database Link Configuration Attributes"
- Section 2.3.1.2.7, "Database Link Configuration Example"

2.3.1.2.1. Providing Suffix Information

Use the ***nsslapd-suffix*** attribute to define the suffix managed by the database link. For example, for the database link to point to the people information for a remote site of the company, enter the following suffix information:

```
nsslapd-suffix: l=Zanzibar,ou=people,dc=example,dc=com
```

The suffix information is stored in the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry.



NOTE

After creating the database link, any alterations to the ***nsslapd-nsslapd-suffix*** attribute are applied only after the server containing the database link is restarted.

2.3.1.2.2. Providing Bind Credentials

For a request from a client application to be chained to a remote server, special bind credentials can be supplied for the client application. This gives the remote server the proxied authorization rights needed to chain operations. Without bind credentials, the database link binds to the remote server as **anonymous**.

Providing bind credentials involves the following steps:

1. On the remote server:
   - Create an administrative user for the database link.

     For information on adding entries, see Chapter 3, *Creating Directory Entries*.
   - Provide proxy access rights for the administrative user created in step 1 on the subtree chained to by the database link.

     For more information on configuring ACIs, see Chapter 13, *Managing Access Control*
2. On the server containing the database link, use **ldapmodify** to provide a user DN for the database link in the ***nsMultiplexorBindDN*** attribute of the **cn=***database_link,***cn=chaining database,cn=plugins,cn=config** entry.

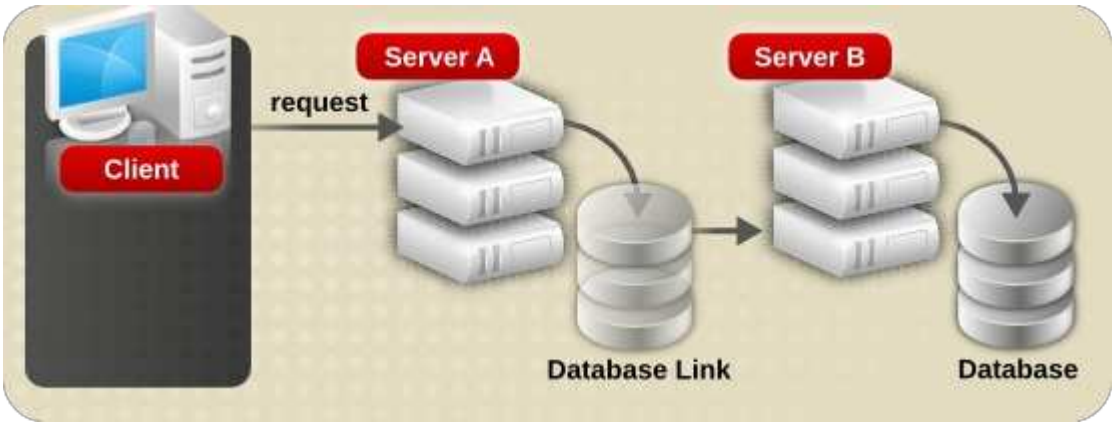
   

   WARNING

   The ***nsMultiplexorBindDN*** cannot be that of the Directory Manager.

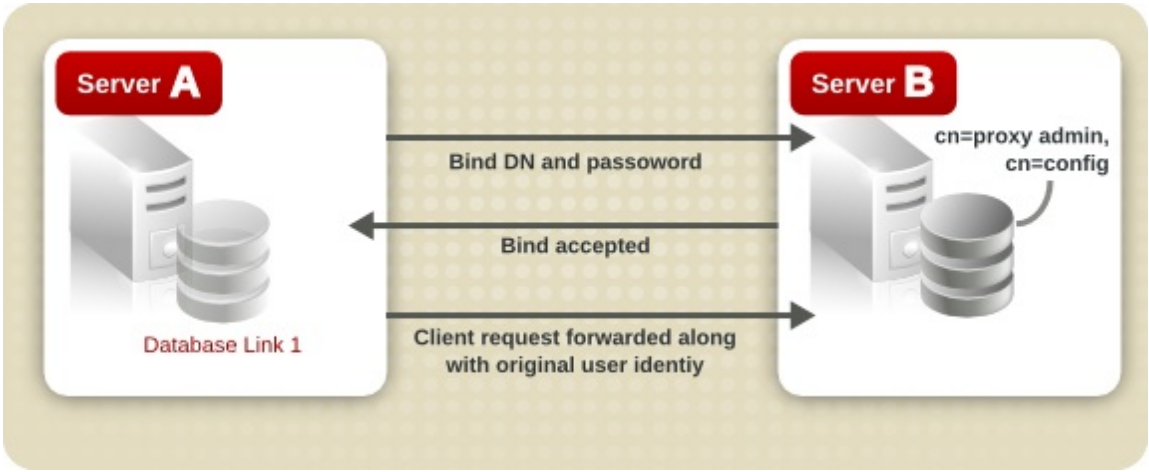
   Use **ldapmodify** to provide a user password for the database link in the ***nsMultiplexorCredentials*** attribute of the **cn=***database_link,***cn=chaining database,cn=plugins,cn=config** entry.

For example, a client application sends a request to Server A. Server A contains a database link that chains the request to a database on Server B.



The database link on Server A binds to Server B using a special user as defined in the ***nsMultiplexorBindDN*** attribute and a user password as defined in the ***nsMultiplexorCredentials*** attribute. In this example, Server A uses the following bind credentials:

```
nsMultiplexorBindDN: cn=proxy admin,cn=config
nsMultiplexorCredentials: secret
```



Server B must contain a user entry corresponding to the ***nsMultiplexorBindDN***, and set the proxy authentication rights for this user. To set the proxy authorization correctly, set the proxy ACI as any other ACI.

WARNING

Carefully examine access controls when enabling chaining to avoid giving access to restricted areas of the directory. For example, if a default proxy ACI is created on a branch, the users that connect using the database link will be able to see all entries below the branch. There may be cases when not all of the subtrees should be viewed by a user. To avoid a security hole, create an additional ACI to restrict access to the subtree.

For more information on ACIs, see Chapter 13, *Managing Access Control*.

NOTE

When a database link is used by a client application to create or modify entries, the attributes ***creatorsName*** and ***modifiersName*** do not reflect the real creator or modifier of the entries. These attributes contain the name of the administrative user granted proxied authorization rights on the remote data server.

2.3.1.2.3. Providing an LDAP URL

On the server containing the database link, identify the remote server that the database link connects with using an *LDAP URL*. Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It takes the form **ldap:**//*server:port*, where the *server* can be a host name, IPv4 address, or IPv6 address.

The URL of the remote server using the ***nsFarmServerURL*** attribute is set in the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry of the configuration file.

```
nsFarmServerURL: ldap://example.com:389/
```

NOTE

Do not forget to use the trailing slash (/) at the end of the URL.

For the database link to connect to the remote server using LDAP over SSL, the LDAP URL of the remote server uses the protocol LDAPS instead of LDAP in the URL and points to the secure port of the server. For example:

```
nsFarmServerURL: ldaps://africa.example.com:636/
```

NOTE

SSL has to be enabled on the local Directory Server and the remote Directory Server to be chained over SSL. For more information on enabling SSL, see Section 7.4, "Setting up TLS/SSL".

When the database link and remote server are configured to communicate using SSL, this does not mean that the client application making the operation request must also communicate using SSL. The client can bind using a normal port.

2.3.1.2.4. Providing a List of Failover Servers

There can be additional LDAP URLs for servers included to use in the case of failure. Add alternate servers to the ***nsFarmServerURL*** attribute, separated by spaces.

```
nsFarmServerURL: ldap://example.com us.example.com:389 africa.example.com:1000/
```

In this sample LDAP URL, the database link first contacts the server **example.com** on the standard port to service an operation. If it does not respond, the database link then contacts the server **us.example.com** on port **389**. If this server fails, it then contacts **africa.example.com** on port **1000**.

2.3.1.2.5. Using Different Bind Mechanisms

The local server can connect to the remote server using several different connection types and authentication mechanisms.

There are three ways that the local server can connect to the remote server:

- Over the standard LDAP port
- Over a dedicated TLS/SSL port
- Using Start TLS, which is a secure connection over a standard port

NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any chaining operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

Ultimately, there are two connection settings. The TLS/SSL option signifies that both of the servers are configured to run and accept connections over TLS/SSL, but there is no separate configuration attribute for enforcing TLS/SSL.

The connection type is identified in the ***nsUseStartTLS*** attribute. When this is **on**, then the server initiates a Start TLS connect over the standard port. If this is **off**, then the server either uses the LDAP port or the TLS/SSL port, depending on what is configured for the remote server in the ***nsFarmServerURL*** attribute.

For example, to use Start TLS:

```
nsUseStartTLS: on
```

For example, to use a standard connection or TLS/SSL connection:

```
nsUseStartTLS: off
```

There are four different methods which the local server can use to authenticate to the farm server.

- *empty*. If there is no bind mechanism set, then the server performs simple authentication and requires the ***nsMultiplexorBindDN*** and ***nsMultiplexorCredentials*** attributes to give the bind information.
- *EXTERNAL*. This uses an SSL certificate to authenticate the farm server to the remote server. Either the farm server URL must be set to the secure URL (**ldaps**) or the ***nsUseStartTLS*** attribute must be set to **on**.

  Additionally, the remote server must be configured to map the farm server's certificate to its bind identity, as described in Section 7.10.2, “Mapping DNs to Certificates”.
- *DIGEST-MD5*. This uses SASL authentication with DIGEST-MD5 encryption. As with simple authentication, this requires the ***nsMultiplexorBindDN*** and ***nsMultiplexorCredentials*** attributes to give the bind information.
- *GSSAPI*. This uses Kerberos-based authentication over SASL.

  The farm server must be configured with a Kerberos keytab, and the remote server must have a defined SASL mapping for the farm server's bind identity. Setting up Kerberos keytabs and SASL mappings is described in Section 7.11, “Setting up SASL Identity Mapping”.

NOTE

SASL connections can be established over standard connections or SSL/TLS connections.

For example:

```
nsBindMechanism: EXTERNAL
```

NOTE

If SASL is used, then the local server must also be configured to chain the SASL and password policy components. Add the components for the database link configuration, as described in Section 2.3.2, “Configuring the Chaining Policy”. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=chaining database,cn=plugins,cn=config
changetype: modify
add: nsActiveChainingComponents
nsActiveChainingComponents: cn=password policy,cn=components,cn=config
-
add: nsActiveChainingComponents
nsActiveChainingComponents: cn=sasl,cn=components,cn=config
^D
```

2.3.1.2.6. Summary of Database Link Configuration Attributes

The following table lists the attributes available for configuring a database link. Some of these attributes were discussed in the earlier sections. All instance attributes are defined in the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry.

Values defined for a specific database link take precedence over the global attribute value.

Table 2.2. Database Link Configuration Attributes

Attributes	Value
nsTransmittedControls [†]	Gives the OID of LDAP controls forwarded by the database link to the remote data server.

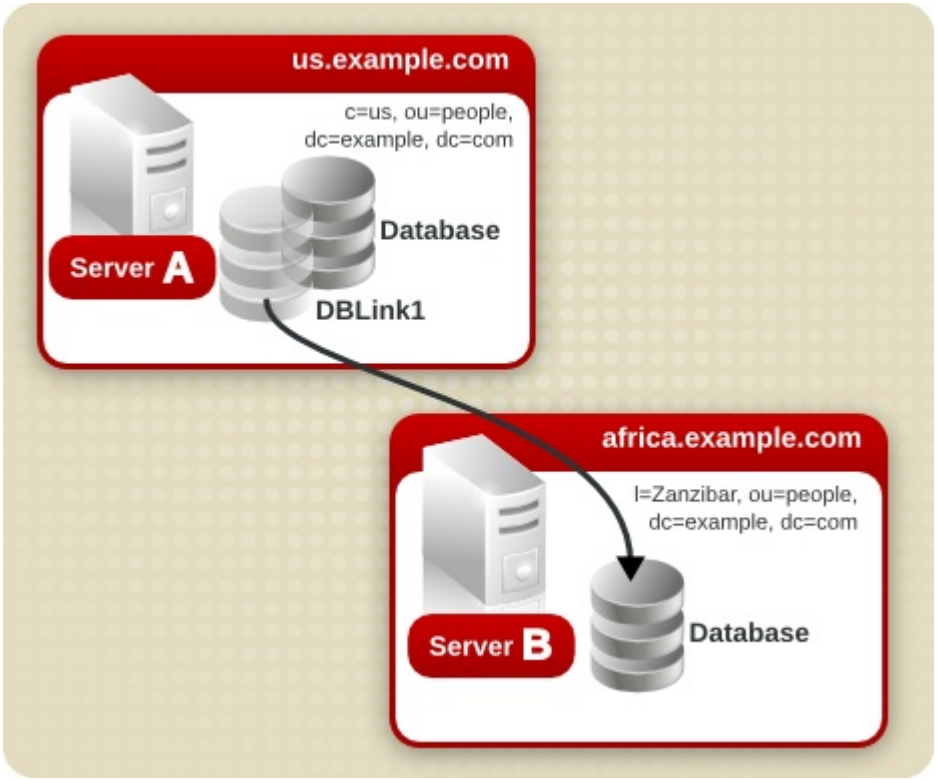
Attributes	Value
nsslapd-suffix	The suffix managed by the database link. Any changes to this attribute after the entry has been created take effect only after the server containing the database link is restarted.
nsslapd-timelimit	Default search time limit for the database link, given in seconds. The default value is **3600** seconds.
nsslapd-sizelimit	Default size limit for the database link, given in number of entries. The default value is **2000** entries.
nsFarmServerURL	Gives the LDAP URL of the remote server (or farm server) that contains the data. This attribute can contain optional servers for failover, separated by spaces. If using cascading chaining, this URL can point to another database link.
nsUseStartTLS	Sets whether to use Start TLS to establish a secure connection over a standard port. The default is **off**, which is used for both simple (standard) connections and TLS/SSL connections.
nsBindMethod	Sets the authentication method to use to authenticate (bind) to the remote server. The default, if this attribute is not given, is to authenticate using a simple bind, requiring the ***nsMultiplexorBindDN*** and ***nsMultiplexorCredentials*** attributes for the bind information. null; this is a simple bind. EXTERNAL (certificate-based); certificate mapping must be enabled for this. DIGEST-MD5 (SASL); this, like a simple bind, requires the bind DN and password. GSSAPI (SASL); this requires the keytab to be configured on the local server and SASL identity mapping on the remote server.
nsMultiplexorBindDN	DN of the administrative entry used to communicate with the remote server. The term ***multiplexor*** in the name of the attribute means the server which contains the database link and communicates with the remote server. This bind DN cannot be the Directory Manager. If this attribute is not specified, the database link binds as **anonymous**.
nsMultiplexorCredentials	Password for the administrative user, given in plain text. If no password is provided, it means that users can bind as **anonymous**. The password is encrypted in the configuration file.
nsCheckLocalACI	Reserved for advanced use only. Controls whether ACIs are evaluated on the database link as well as the remote data server. Takes the values **on** or **off**. Changes to this attribute occur only after the server has been restarted. The default value is **off**.
nsProxiedAuthorization	Reserved for advanced use only. Disables proxied authorization. A value of **off** means proxied authorization is disabled. The default value is **on**.

Attributes	Value
nsActiveChainingComponents[†]	Lists the components using chaining. A component is any functional unit in the server. The value of this attribute in the database link instance overrides the value in the global configuration attribute. To disable chaining on a particular database instance, use the value **none**. The default policy is not to allow chaining. For more information, see Section 2.3.2.1, "Chaining Component Operations".
nsReferralOnScopedSearch	Controls whether referrals are returned by scoped searches. This attribute is for optimizing the directory because returning referrals in response to scoped searches is more efficient. Takes the values **on** or **off**. The default value is **off**.
nsHopLimit	Maximum number of times a request can be forwarded from one database link to another. The default value is **10**.

[†] Can be both a global and instance attribute. This global configuration attribute is located in the **cn=config,cn=chaining database,cn=plugins,cn=config** entry. The global attributes are dynamic, meaning any changes made to them automatically take effect on all instances of the database link within the directory.

2.3.1.2.7. Database Link Configuration Example

Suppose a server within the **us.example.com** domain contains the subtree **l=Walla Walla,ou=people,dc=example,dc=com** on a database and that operation requests for the **l=Zanzibar,ou=people,dc=example,dc=com** subtree should be chained to a different server in the **africa.example.com** domain.



1. Run **ldapmodify** to add a database link to Server A:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Specify the configuration information for the database link:

```
dn: cn=DBLink1,cn=chaining database,cn=plugins,cn=config
changetype: add
objectclass: top
```

```
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: c=africa,ou=people,dc=example,dc=com
nsfarmserverurl: ldap://africa.example.com:389/
nsMultiplexorBindDN: cn=proxy admin,cn=config
nsMultiplexorCredentials: secret
cn: DBLink1

dn: cn=c=africa\,ou=people\,dc=example\,dc=com,cn=mapping tree,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsMappingTree
nsslapd-state: backend
nsslapd-backend: DBLink1
nsslapd-parent-suffix: ou=people,dc=example,dc=com
cn: c=africa\,ou=people\,dc=example\,dc=com
```

In the first entry, the ***nsslapd-suffix*** attribute contains the suffix on Server B to which to chain from Server A. The ***nsFarmServerURL*** attribute contains the LDAP URL of Server B.

The second entry creates a new suffix, allowing the server to route requests made to the new database link. The ***cn*** attribute contains the same suffix specified in the ***nsslapd-suffix*** attribute of the database link. The ***nsslapd-backend*** attribute contains the name of the database link. The ***nsslapd-parent-suffix*** attribute specifies the parent of this new suffix, **ou=people,dc=example,dc=com**.

3. Create an administrative user on Server B, as follows:

```
dn: cn=proxy admin,cn=config
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: proxy admin
sn: proxy admin
userPassword: secret
description: Entry for use by database links
```

WARNING

Do not use the Directory Manager user as the proxy administrative user on the remote server. This creates a security hole.

4. Add the following proxy authorization ACI to the **l=Zanzibar,ou=people,dc=example,dc=com** entry on Server B:

```
aci: (targetattr = "*")(version 3.0; acl "Proxied authorization
    for database links"; allow (proxy) userdn = "ldap:///cn=proxy
    admin,cn=config";)
```

This ACI gives the proxy admin user read-only access to the data contained on the remote server within the **l=Zanzibar,ou=people,dc=example,dc=com** subtree only.

NOTE

When a user binds to a database link, the user's identity is sent to the remote server. Access controls are always evaluated on the remote server. For the user to modify or write data successfully to the remote server, set up the correct access controls on the remote server. For more information about how access controls are evaluated in the context of chained operations, see Section 2.3.6, “Database Links and Access Control Evaluation” .

2.3.2. Configuring the Chaining Policy

These procedures describe configuring how Directory Server chains requests made by client applications to Directory Servers that contain database links. This chaining policy applies to all database links created on Directory Server.

2.3.2.1. Chaining Component Operations

A component is any functional unit in the server that uses internal operations. For example, plug-ins are considered to be components, as are functions in the front-end. However, a plug-in may actually be comprised of multiple components (for example, the ACI plug-in).

Some components send internal LDAP requests to the server, expecting to access local data only. For such components, control the chaining policy so that the components can complete their operations successfully. One example is the certificate verification function. Chaining the LDAP request made by the function to check certificates implies that the remote server is trusted. If the remote server is not trusted, then there is a security problem.

By default, all internal operations are not chained and no components are allowed to chain, although this can be overridden.

Additionally, an ACI must be created on the remote server to allow the specified plug-in to perform its operations on the remote server. The ACI must exist in the *suffix* assigned to the database link.

The following table lists component names, the potential side-effects of allowing them to chain internal operations, and the permissions they need in the ACI on the remote server:

Table 2.3. Components Allowed to Chain

Component Name	Description	Permissions
ACI plug-in	This plug-in implements access control. Operations used to retrieve and update ***ACI*** attributes are not chained because it is not safe to mix local and remote ACI attributes. However, requests used to retrieve user entries may be chained by setting the chaining components attribute, *nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config*.	Read, search, and compare
Resource limit component	This component sets server limits depending on the user bind DN. Resource limits can be applied on remote users if the resource limitation component is allowed to chain. To chain resource limit component operations, add the chaining component attribute, *nsActiveChainingComponents: cn=resource limits,cn=components,cn=config*.	Read, search, and compare
Certificate-based authentication checking component	This component is used when the external bind method is used. It retrieves the user certificate from the database on the remote server. Allowing this component to chain means certificate-based authentication can work with a database link. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=certificate-based authentication,cn=components,cn=config*.	Read, search, and compare

Component Name	Description	Permissions
Password policy component	This component is used to allow SASL binds to the remote server. Some forms of SASL authentication require authenticating with a user name and password. Enabling the password policy allows the server to verify and implement the specific authentication method requested and to apply the appropriate password policies. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=password policy,cn=components,cn=config*.	Read, search, and compare
SASL component	This component is used to allow SASL binds to the remote server. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=password policy,cn=components,cn=config*.	Read, search, and compare
Referential Integrity plug-in	This plug-in ensures that updates made to attributes containing DNs are propagated to all entries that contain pointers to the attribute. For example, when an entry that is a member of a group is deleted, the entry is automatically removed from the group. Using this plug-in with chaining helps simplify the management of static groups when the group members are remote to the static group definition. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=referential integrity postoperation,cn=plugins,cn=config*.	Read, write, search, and compare
Attribute Uniqueness plug-in	This plug-in checks that all the values for a specified attribute are unique (no duplicates). If this plug-in is chained, it confirms that attribute values are unique even on attributes changed through a database link. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=attribute uniqueness,cn=plugins,cn=config*	Read, search, and compare
Roles component	This component chains the roles and roles assignments for the entries in a database. Chaining this component maintains the roles even on chained databases. To chain this component's operations, add the chaining component attribute, *nsActiveChainingComponents: cn=roles,cn=components,cn=config*.	Read, write, search, and compare

NOTE

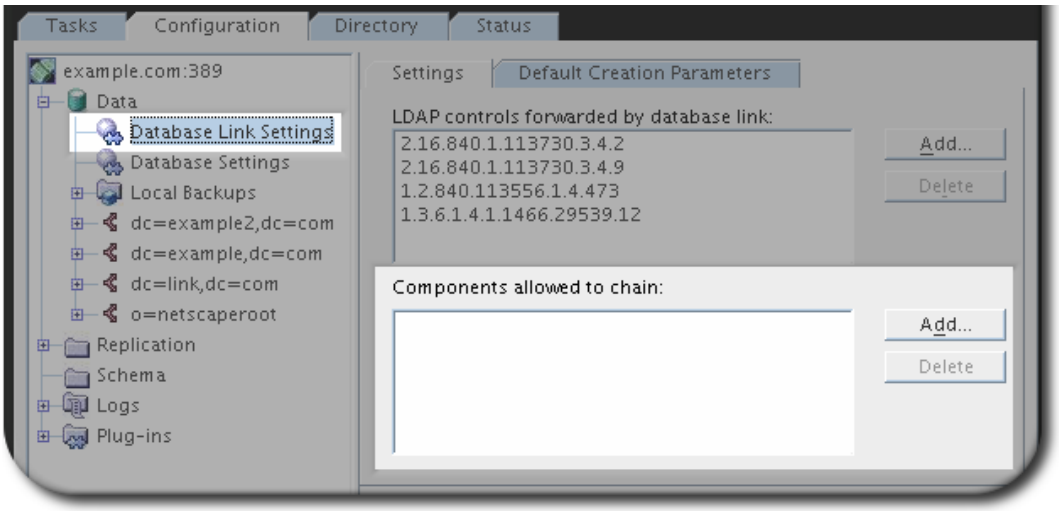
The following components cannot be chained:

- Roles plug-in
- Password policy component
- Replication plug-ins
- Referential Integrity plug-in

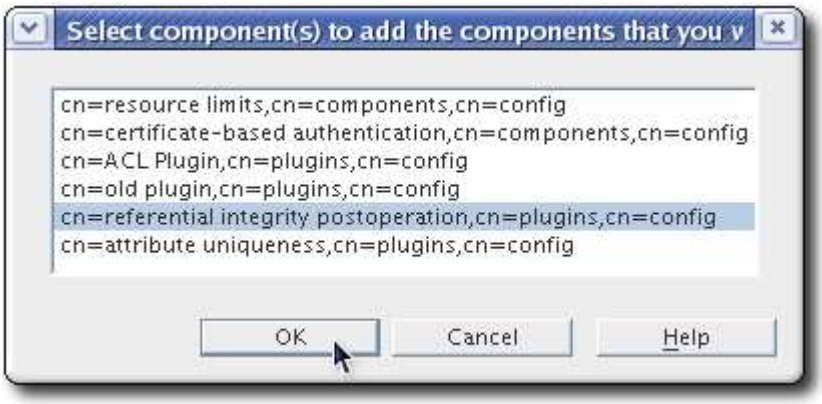
When enabling the Referential Integrity plug-in on servers issuing chaining requests, be sure to analyze performance, resource, and time needs as well as integrity needs. Integrity checks can be time-consuming and draining on memory and CPU. For further information on the limitations surrounding ACIs and chaining, see Section 13.1.4, "ACI Limitations".

2.3.2.1.1. Chaining Component Operations Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand **Data** in the left pane, and click **Database Link Settings**.
3. Select the **Settings** tab in the right window.



4. Click the **Add** button in the **Components allowed to chain** section.
5. Select the component to chain from the list, and click **OK**.



6. Restart the server in order for the change to take effect.

After allowing the component to chain, create an ACI in the suffix on the remote server to which the operation will be chained. For example, this creates an ACI for the Referential Integrity plug-in:

```
aci: (targetattr "*")(target="ldap:///ou=customers,l=us,dc=example,dc=com")
    (version 3.0; acl "RefInt Access for chaining"; allow
    (read,write,search,compare) userdn = "ldap:///cn=referential integrity
```

```
    postoperation,cn=plugins,cn=config";)
```

2.3.2.1.2. Chaining Component Operations from the Command Line

1. Specify components to include in chaining using the ***nsActiveChainingComponents*** attribute in the **cn=config,cn=chaining database,cn=plugins,cn=config** entry of the configuration file.

   For example, to allow the referential integrity component to chain operations, add the following to the database link configuration file:

   ```
   nsActiveChainingComponents: cn=referential integrity postoperation,cn=components,cn=config
   ```

   See Table 2.3, "Components Allowed to Chain" for a list of the components which can be chained.

2. Restart the server for the change to take effect.

   ```
   service dirsrv restart instance
   ```

3. Create an ACI in the suffix on the remote server to which the operation will be chained. For example, this creates an ACI for the Referential Integrity plug-in:

   ```
   aci: (targetattr "*")(target="ldap:///ou=customers,l=us,dc=example,dc=com")
       (version 3.0; acl "RefInt Access for chaining"; allow
       (read,write,search,compare) userdn = "ldap:///cn=referential
       integrity postoperation,cn=plugins,cn=config";)
   ```

2.3.2.2. Chaining LDAP Controls

It is possible to *not* chain operation requests made by LDAP controls. By default, requests made by the following controls are forwarded to the remote server by the database link:

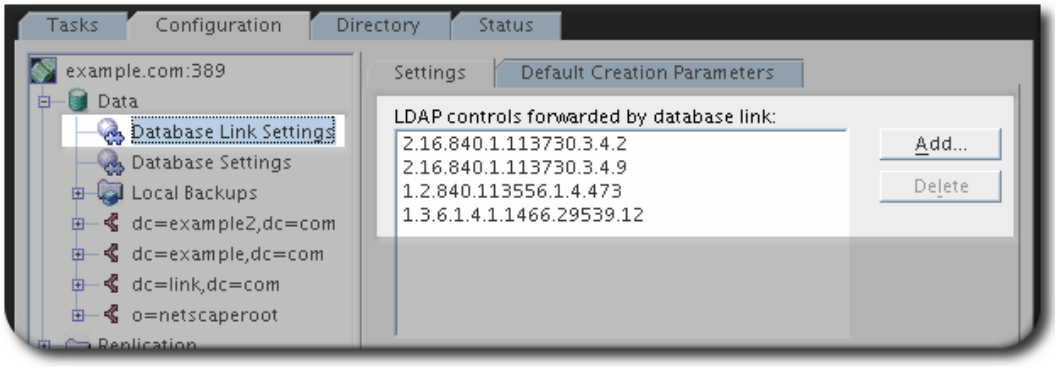
- *Virtual List View (VLV).* This control provides lists of parts of entries rather than returning all entry information.
- *Server-side sorting.* This control sorts entries according to their attribute values, usually using a specific matching rule.
- *Dereferencing.* This control tracks back over references in entry attributes in a search and pulls specified attribute information from the referenced entry and returns it with the rest of the search results.
- *Managed DSA.* This controls returns smart referrals as entries, rather than following the referral, so the smart referral itself can be changed or deleted.
- *Loop detection.* This control keeps track of the number of times the server chains with another server. When the count reaches the configured number, a loop is detected, and the client application is notified. For more information about using this control, see Section 2.4.4, "Detecting Loops".

NOTE

Server-side sorting and VLV controls are supported only when a client application request is made to a single database. Database links cannot support these controls when a client application makes a request to multiple databases.

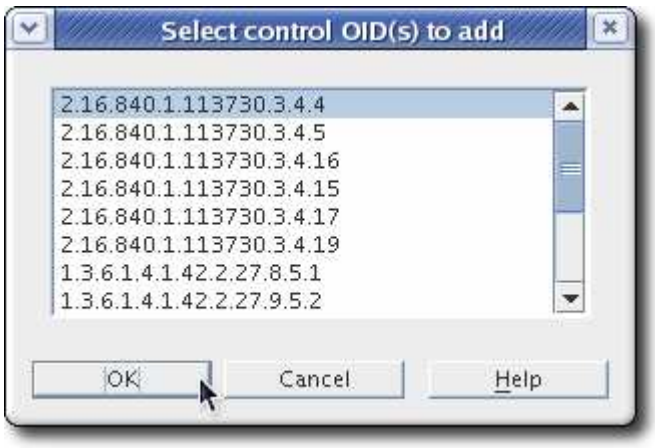
2.3.2.2.1. Chaining LDAP Controls Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder in the left pane, and click **Database Link Settings**.
3. Select the **Settings** tab in the right window.



4. Click the **Add** button in the **LDAP Controls forwarded by the database link** section to add an LDAP control to the list.

5. Select the OID of a control to add to the list, and click **OK**.



2.3.2.2.2. Chaining LDAP Controls from the Command Line

To chain controls, alter the controls that the database link forwards by changing the ***nsTransmittedControls*** attribute of the **cn=config,cn=chaining database,cn=plugins,cn=config** entry. For example, to forward the virtual list view control, add the following to the database link entry in the configuration file:

```
nsTransmittedControls: 2.16.840.1.113730.3.4.9
```

In addition, if clients of the Directory Server create their own controls and their operations should be chained to remote servers, add the OID of the custom control to the ***nsTransmittedControls*** attribute.

The LDAP controls which can be chained and their OIDs are listed in the following table:

Table 2.4. LDAP Controls and Their OIDs

Control Name	OID
Virtual list view (VLV)	2.16.840.1.113730.3.4.9
Server-side sorting	1.2.840.113556.1.4.473
Managed DSA	2.16.840.1.113730.3.4.2
Loop detection	1.3.6.1.4.1.1466.29539.12
Dereferencing searches	1.3.6.1.4.1.4203.666.5.16

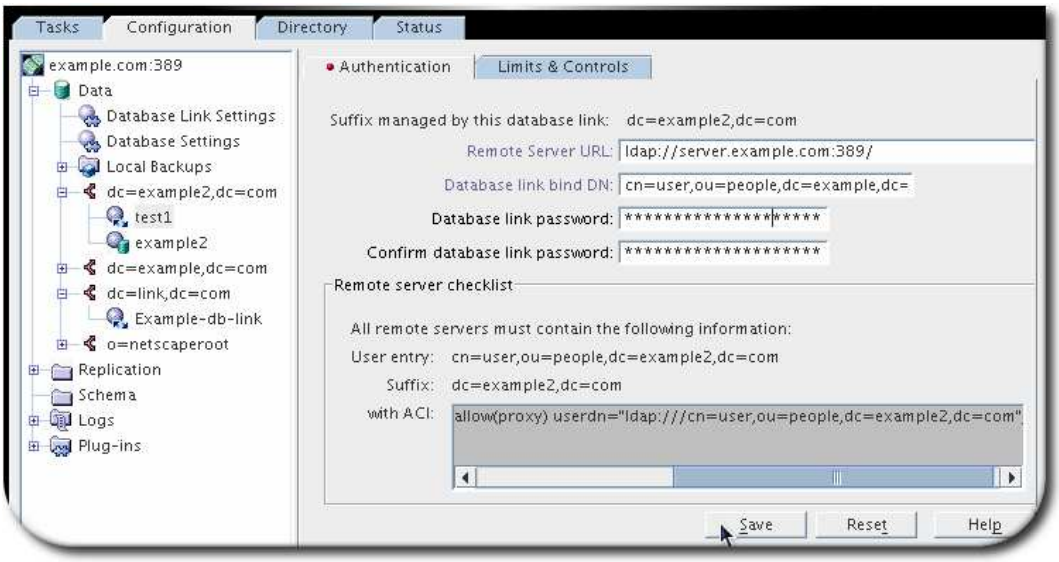
2.3.3. Maintaining Database Links

All of the information for the database link for the connection to the remote server.

1. In the Directory Server Console, select the **Configuration** tab.

2. In the left pane, expand the **Data** folder, and select the database link under the suffix.

3. In the right navigation pane, click the **Authentication** tab.



4. Change the connection information.

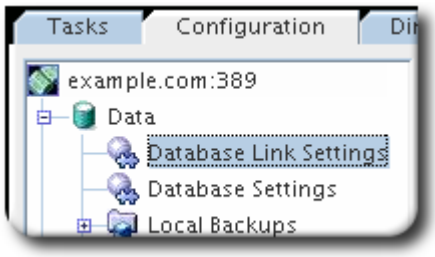
   - The LDAP URL for the remote server.[]
   - The bind DN and password used by the database link to bind to the remote server.

2.3.4. Configuring Database Link Defaults

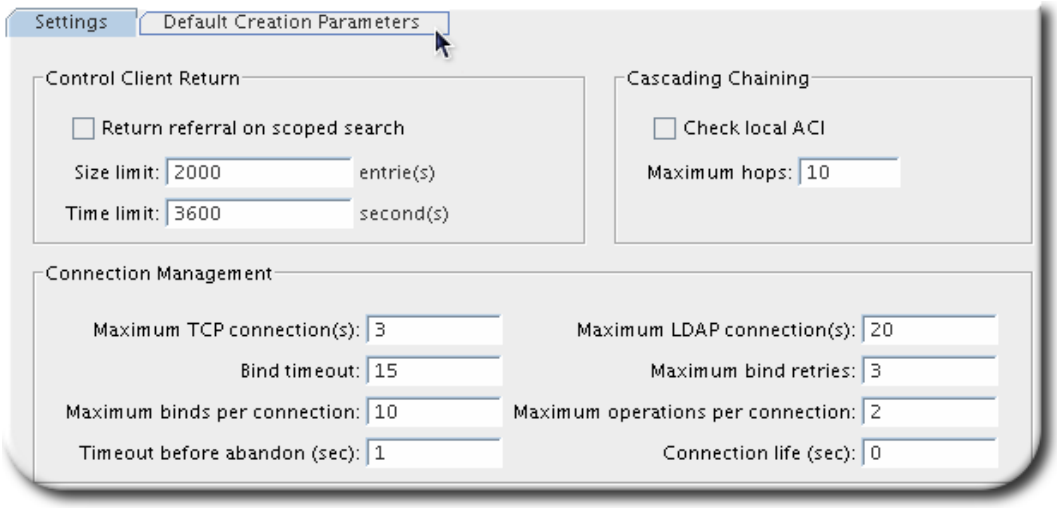
Configuring the default settings for database links defines the settings used for cascading chaining (the number of hops allowed for a client request), the connection rules for the remote server, and how the server responds to client requests.

1. Select the **Configuration** tab.

2. Expand the **Data** folder in the left pane, and click **Database Link Settings**. Open the **Default Creation Parameters** tab.



3. Fill in the new configuration parameters.



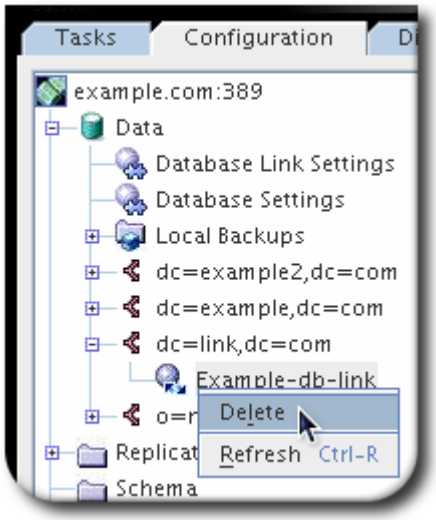
NOTE

Changes made to the default settings of a database link are not applied retroactively. Only the database links created after changes are made to the default settings will reflect the changes.

2.3.5. Deleting Database Links

To delete a database link, right-click the database link, and select **Delete** from the pop-up menu. Confirm the delete when prompted.

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left navigation pane, open the suffix and select the database link to delete.
3. Right-click the database link, and select **Delete** from the menu.



2.3.6. Database Links and Access Control Evaluation

When a user binds to a server containing a database link, the database link sends the user's identity to the remote server. Access controls are always evaluated on the remote server. Every LDAP operation evaluated on the remote server uses the original identity of the client application passed using the proxied authorization control. Operations succeed on the remote server only if the user has the correct access controls on the subtree contained on the remote server. This requires adding the usual access controls to the remote server with a few restrictions:

- Not all types of access control can be used.

  For example, role-based or filter-based ACIs need access to the user entry. Because the data are accessed through database links, only the data in the proxy control can be verified. Consider designing the directory in a way that ensures the user entry is located in the same database as the user's data.

- All access controls based on the IP address or DNS domain of the client may not work since the original domain of the client is lost during chaining. The remote server views the client application as being at the same IP address and in the same DNS domain as the database link.

NOTE

Directory Server supports both IPv4 and IPv6 IP addresses.

The following restrictions apply to the ACIs used with database links:

- ACIs must be located with any groups they use. If the groups are dynamic, all users in the group must be located with the ACI and the group. If the group is static, it links to remote users.
- ACIs must be located with any role definitions they use and with any users intended to have those roles.
- ACIs that link to values of a user's entry (for example, **userattr** subject rules) will work if the user is remote.

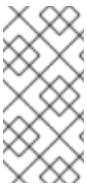
Though access controls are always evaluated on the remote server, they can also be evaluated on both the server containing the database link and the remote server. This poses several limitations:

- During access control evaluation, contents of user entries are not necessarily available (for example, if the access control is evaluated on the server containing the database link and the entry is located on a remote server).

  For performance reasons, clients cannot do remote inquiries and evaluate access controls.

- The database link does not necessarily have access to the entries being modified by the client application.

  When performing a modify operation, the database link does not have access to the full entry stored on the remote server. If performing a delete operation, the database link is only aware of the entry's DN. If an access control specifies a particular attribute, then a delete operation will fail when being conducted through a database link.



NOTE

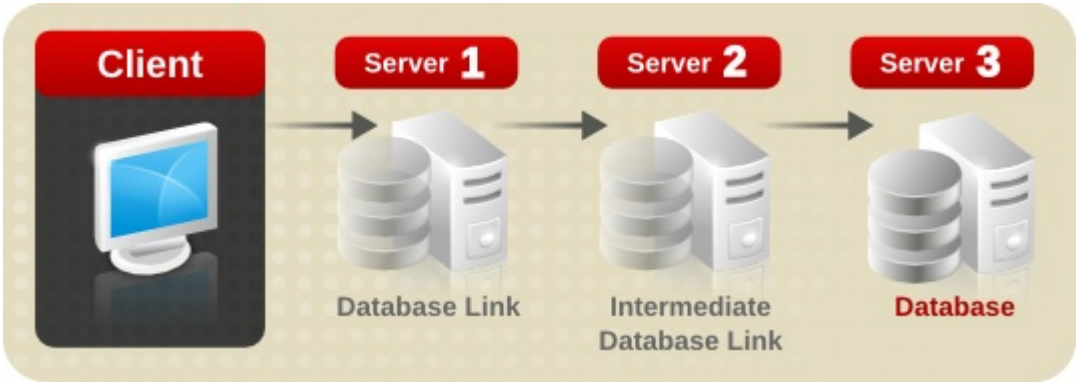
By default, access controls set on the server containing the database link are not evaluated. To override this default, use the ***nsCheckLocalACI*** attribute in the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry. However, evaluating access controls on the server containing the database link is not recommended except with cascading chaining.

2.4. CONFIGURING CASCADING CHAINING

The database link can be configured to point to another database link, creating a cascading chaining operation. A cascading chain occurs any time more than one hop is required to access all of the data in a directory tree.

2.4.1. Overview of Cascading Chaining

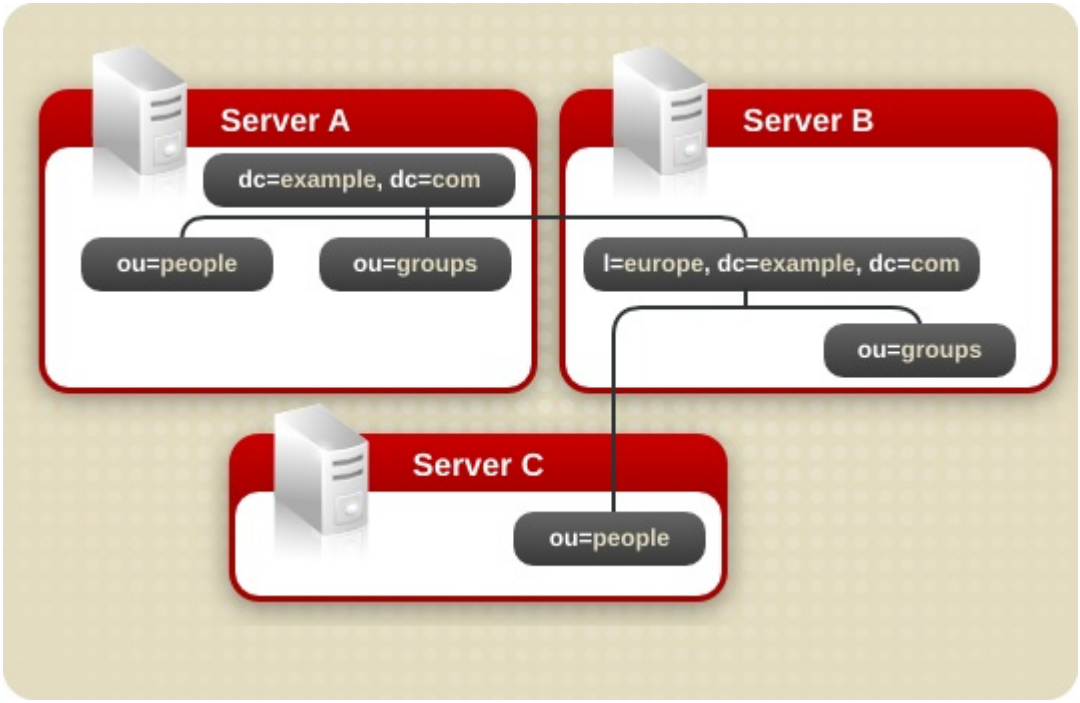
Cascading chaining occurs when more than one hop is required for the directory to process a client application's request.



The client application sends a modify request to Server 1. Server one contains a database link that forwards the operation to Server 2, which contains another database link. The database link on Server 2 forwards the operations to server three, which contains the data the clients wants to modify in a database. Two hops are required to access the piece of data the client want to modify.

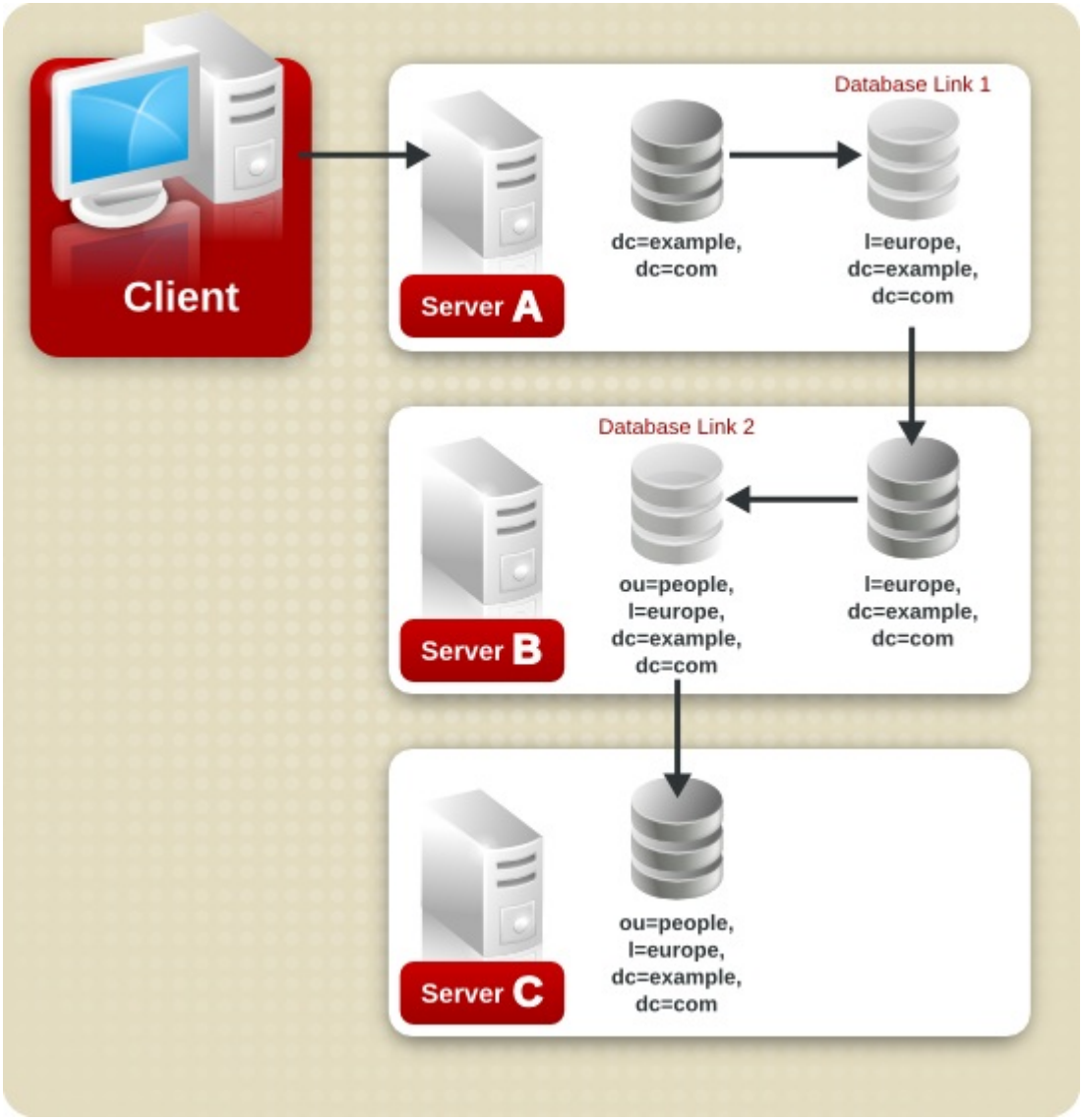
During a normal operation request, a client binds to the server, and then any ACIs applying to that client are evaluated. With cascading chaining, the client bind request is evaluated on Server 1, but the ACIs applying to the client are evaluated only after the request has been chained to the destination server, in the above example Server 2.

For example, on Server A, a directory tree is split:



The root suffix **dc=example,dc=comand** the **ou=people** and **ou=groups** sub suffixes are stored on Server A. The **l=europe,dc=example,dc=com** and **ou=groups** suffixes are stored in on Server B, and the **ou=people** branch of the **l=europe,dc=example,dc=com** suffix is stored on Server C.

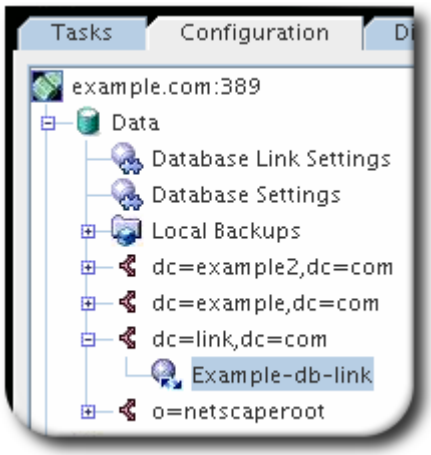
With cascading configured on servers A, B, and C, a client request targeted at the **ou=people,l=europe,dc=example,dc=com** entry would be routed by the directory as follows:



First, the client binds to Server A and chains to Server B using Database Link 1. Then Server B chains to the target database on Server C using Database Link 2 to access the data in the **ou=people,l=europe,dc=example,dc=com** branch. Because at least two hops are required for the directory to service the client request, this is considered a cascading chain.

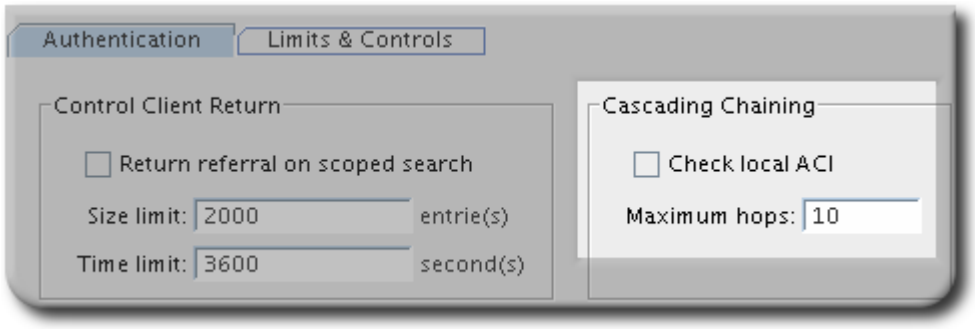
2.4.2. Configuring Cascading Chaining Using the Console

1. Select the **Configuration** tab. Expand the **Data** folder in the left pane, and select the suffix, then the database link.



2. Click the **Limits and Controls** tab in the right navigation pane.

3. Select the **Check local ACI** check box to enable the evaluation of local ACIs on the intermediate database links involved in the cascading chain. Selecting this check box may require adding the appropriate local ACIs to the database link.



4. Enter the maximum number of times a database link can point to another database link in the **Maximum hops** field.

   By default, the maximum is ten hops. After ten hops, a loop is detected by the server, and an error is returned to the client application.

2.4.3. Configuring Cascading Chaining from the Command Line

To configure a cascade of database links through the command line:

1. Point one database link to the URL of the server containing the intermediate database link.

   To create a cascading chain, the ***nsFarmServerURL*** attribute of one database link must contain the URL of the server containing another database link. Suppose the database link on the server called **example1.com** points to a database link on the server called **africa.example.com**. For example, the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry of the database link on Server 1 would contain the following:

   ```
   nsFarmServerURL: ldap://africa.example.com:389/
   ```

2. Configure the intermediate database link or links (in the example, Server 2) to transmit the Proxy Authorization Control.

   By default, a database link does not transmit the Proxy Authorization Control. However, when one database link contacts another, this control is used to transmit information needed by the final destination server. The intermediate database link needs to transmit this control. To configure the database link to transmit the proxy authorization control, add the following to the **cn=config,cn=chaining database,cn=plugins,cn=config** entry of the intermediate database link:

   ```
   nsTransmittedControls: 2.16.840.1.113730.3.4.12
   ```

   The OID value represents the Proxy Authorization Control. For more information about chaining LDAP controls, see Section 2.3.2.2, “Chaining LDAP Controls”.

3. Create a proxy administrative user ACI on all intermediate database links.

   The ACI must exist on the server that contains the intermediate database link that checks the rights of the first database link before translating the request to another server. For example, if Server 2 does not check the credentials of Server 1, then anyone could bind as **anonymous** and pass a proxy authorization control allowing them more administrative privileges than appropriate. The proxy ACI prevents this security breach.

   1. Create a database, if one does not already exist, on the server containing the intermediate database link. This database will contain the admin user entry and the ACI. For information about creating a database, see Section 2.2.1, “Creating Databases”.

   2. Create an entry that corresponds to the administrative user in the database.

   3. Create an ACI for the administrative user that targets the appropriate suffix. This ensures the administrator has access only to the suffix of the database link. For example:

      ```
      aci: (targetattr = "*")(version 3.0; acl "Proxied authorization for database links";
          allow (proxy) userdn = "ldap:///cn=proxy admin,cn=config";)
      ```

This ACI is like the ACI created on the remote server when configuring simple chaining.

WARNING

Carefully examine access controls when enabling chaining to avoid giving access to restricted areas of the directory. For example, if a default proxy ACI is created on a branch, the users that connect through the database link will be able to see all entries below the branch. There may be cases when not all of the subtrees should be viewed by a user. To avoid a security hole, create an additional ACI to restrict access to the subtree.

4. Enable local ACI evaluation on all intermediate database links.

   To confirm that the proxy administrative ACI is used, enable evaluation of local ACIs on all intermediate database links involved in chaining. Add the following attribute to the **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry of each intermediate database link:

   ```
   nsCheckLocalACI: on
   ```

   Setting this attribute to **on** in the **cn=default instance config,cn=chaining database,cn=plugins,cn=config** entry means that all new database link instances will have the ***nsCheckLocalACI*** attribute set to **on** in their **cn=***database_link,* **cn=chaining database,cn=plugins,cn=config** entry.

5. Create client ACIs on all intermediate database links and the final destination database.

   Because local ACI evaluation is enabled, the appropriate client application ACIs must be created on all intermediate database links, as well as the final destination database. To do this on the intermediate database links, first create a database that contains a suffix that represents a root suffix of the final destination suffix.

   For example, if a client request made to the **c=africa,ou=people,dc=example,dc=com** suffix is chained to a remote server, all intermediate database links need to contain a database associated with the **dc=example,dc=com** suffix.

   Add any client ACIs to this superior suffix entry. For example:

   ```
   aci: (targetattr = "*")(version 3.0; acl "Client authentication for database link users";
        allow (all) userdn = "ldap:///uid=* ,cn=config";)
   ```

   This ACI allows client applications that have a ***uid*** in the **cn=config** entry of Server 1 to perform any type of operation on the data below the **ou=people,dc=example,dc=com** suffix on server three.

2.4.4. Detecting Loops

An LDAP control included with Directory Server prevents loops. When first attempting to chain, the server sets this control to be the maximum number of hops, or chaining connections, allowed. Each subsequent server decrements the count. If a server receives a count of **0**, it determines that a loop has been detected and notifies the client application.

The number of hops allowed is defined using the ***nsHopLimit*** attribute. If not specified, the default value is **10**.

To use the control, add the following OID to the ***nsTransmittedControl*** attribute in the **cn=config,cn=chaining database,cn=plugins,cn=config** entry:

```
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

If the control is not present in the configuration file of each database link, loop detection will not be implemented.

2.4.5. Summary of Cascading Chaining Configuration Attributes

The following table describes the attributes used to configure intermediate database links in a cascading chain:

Table 2.5. Cascading Chaining Configuration Attributes

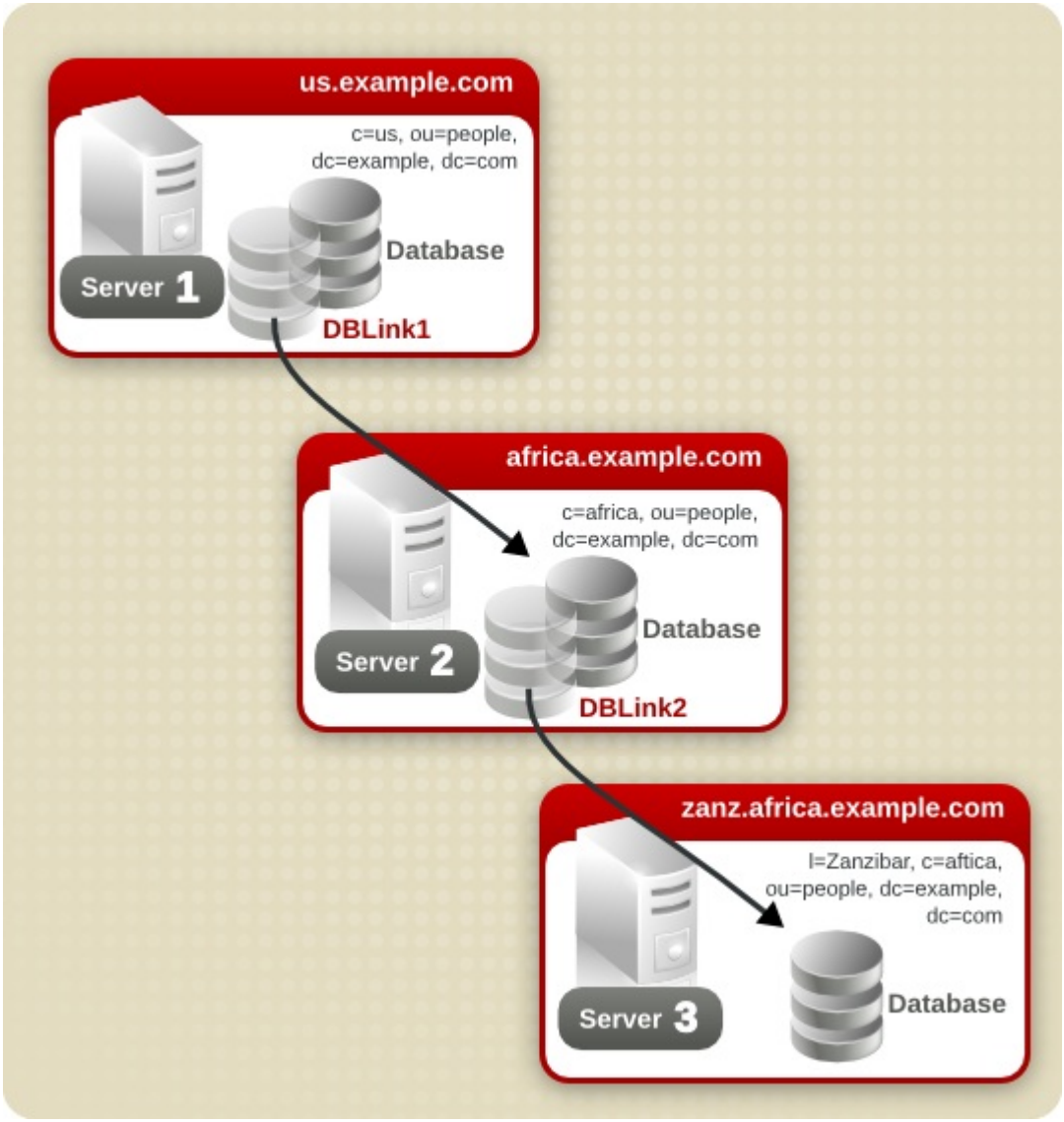
Attribute	Description
nsFarmServerURL	URL of the server containing the next database link in the cascading chain.

nsTransmittedControls	Enter the following OIDs to the database links involved in the cascading chain:
	`nsTransmittedControls: 2.16.840.1.113730.3.4.12`
	`nsTransmittedControls: 1.3.6.1.4.1.1466.29539.12`
	The first OID corresponds to the Proxy Authorization Control. The second OID corresponds to the Loop Detection Control.

aci	This attribute must contain the following ACI: `aci: (targetattr = "*")(version 3.0; acl "Proxied authorization for database links"; allow (proxy) userdn = "ldap:///cn=proxy admin,cn=config";)`
nsCheckLocalACI	To enable evaluation of local ACIs on all database links involved in chaining, turn local ACI evaluation on, as follows: `nsCheckLocalACI: on`

2.4.6. Cascading Chaining Configuration Example

To create a cascading chain involving three servers as in the diagram below, the chaining components must be configured on all three servers.



- Section 2.4.6.1, “Configuring Server One”
- Section 2.4.6.2, “Configuring Server Two”
- Section 2.4.6.3, “Configuring Server Three”

2.4.6.1. Configuring Server One

1. Run **ldapmodify** and specify the configuration information for the database link, DBLink1, on Server 1:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=DBLink1,cn=chaining database,cn=plugins,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: c=africa,ou=people,dc=example,dc=com
nsfarmserverurl: ldap://africa.example.com:389/
nsMultiplexorBindDN: cn=server1 proxy admin,cn=config
nsMultiplexorCredentials: secret
cn: DBLink1
nsCheckLocalACI:off

dn: cn=c=africa\,ou=people\,dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: nsMappingTree
nsslapd-state: backend
```

```
nsslapd-backend: DBLink1
nsslapd-parent-suffix: ou=people,dc=example,dc=com
cn: c=africa\,ou=people\,dc=example\,dc=com
```

The first section creates the entry associated with **DBLink1**. The second section creates a new suffix, allowing the server to direct requests made to the database link to the correct server. The ***nsCheckLocalACI*** attribute does not need to be configured to check local ACIs, as this is only required on the database link, **DBLink2**, on Server 2.

2. To implement loop detection, to specify the OID of the loop detection control in the ***nsTransmittedControl*** attribute stored in **cn=config,cn=chaining database,cn=plugins,cn=config** entry on Server 1.

```
dn: cn=config,cn=chaining database,cn=plugins,cn=config
changetype: modify
add: nsTransmittedControl
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

As the ***nsTransmittedControl*** attribute is usually configured by default with the loop detection control OID **1.3.6.1.4.1.1466.29539.12** value, it is wise to check beforehand whether it already exists. If it does exist, this step is not necessary.

2.4.6.2. Configuring Server Two

1. Create a proxy administrative user on Server 2. This administrative user will be used to allow Server 1 to bind and authenticate to Server 2. It is useful to choose a proxy administrative user name which is specific to Server 1, as it is the proxy administrative user which will allow server *one* to bind to Server 2. Create the proxy administrative user, as follows:

```
dn: cn=server1 proxy admin,cn=config
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: server1 proxy admin
sn: server1 proxy admin
userPassword: secret
description: Entry for use by database links
```

WARNING

Do not use the Directory Manager or Administrator ID user as the proxy administrative user on the remote server. This creates a security hole.

2. Configure the database link, **DBLink2**, on Server 2:

```
dn: cn=DBLink2,cn=chaining database,cn=plugins,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: l=Zanzibar,c=africa,ou=people,dc=example,dc=com
nsfarmserverurl: ldap://zanz.africa.example.com:389/
nsMultiplexorBindDN: cn=server2 proxy admin,cn=config
nsMultiplexorCredentials: secret
cn: DBLink2
nsCheckLocalACI:on

dn: cn=l=Zanzibar\,c=africa\,ou=people\,dc=example\,dc=com,cn=mapping tree,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsMappingTree
nsslapd-state: backend
nsslapd-backend: DBLink2
nsslapd-parent-suffix: c=africa,ou=people,dc=example,dc=com
cn: l=Zanzibar\,c=africa\,ou=people\,dc=example\,dc=com
```

Since database link DBLink2 is the intermediate database link in the cascading chaining configuration, set the ***nsCheckLocalACI*** attribute to **on** to allow the server to check whether it should allow the client and proxy administrative user access to the database link.

3. The database link on Server 2 must be configured to transmit the proxy authorization control and the loop detection control. To implement the proxy authorization control and the loop detection control, specify both corresponding OIDs. Add the following information to the **cn=config,cn=chaining database,cn=plugins,cn=config** entry on Server 2:

```
dn: cn=config,cn=chaining database,cn=plugins,cn=config
changetype: modify
add: nsTransmittedControl
nsTransmittedControl: 2.16.840.1.113730.3.4.12
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

nsTransmittedControl: 2.16.840.1.113730.3.4.12 is the OID for the proxy authorization control. **nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12** is the or the loop detection control.

Check beforehand whether the loop detection control is already configured, and adapt the above command accordingly.

4. Configure the ACIs. On Server 2, ensure that a suffix exists above the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** suffix, so that the following actions are possible:

   - Add the database link suffix
   - Add a local proxy authorization ACI to allow Server 1 to connect using the proxy authorization administrative user created on Server 2
   - Add a local client ACI so the client operation succeeds on Server 2, and it can be forwarded to server three. This local ACI is needed because local ACI checking is turned on for the **DBLink2** database link.

   Both ACIs will be placed on the database that contains the **c=africa,ou=people,dc=example,dc=com** suffix.

NOTE

To create these ACIs, the database corresponding to the **c=africa,ou=people,dc=example,dc=com** suffix must already exist to hold the entry. This database needs to be associated with a suffix above the suffix specified in the ***nsslapd-suffix*** attribute of each database link. That is, the suffix on the final destination server should be a sub suffix of the suffix specified on the intermediate server.

   1. Add the local proxy authorization ACI to the **c=africa,ou=people,dc=example,dc=com** entry:

```
aci:(targetattr="*")(target="l=Zanzibar,c=africa,ou=people,dc=example,dc=com")
    (version 3.0; acl "Proxied authorization for database links"; allow (proxy)
    userdn = "ldap:///cn=server1 proxy admin,cn=config";)
```

   2. Then add the local client ACI that will allow the client operation to succeed on Server 2, given that ACI checking is turned on. This ACI is the same as the ACI created on the destination server to provide access to the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** branch. All users within **c=us,ou=people,dc=example,dc=com** may need to have update access to the entries in **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** on server three. Create the following ACI on Server 2 on the **c=africa,ou=people,dc=example,dc=com** suffix to allow this:

```
aci:(targetattr="*")(target="l=Zanzibar,c=africa,ou=people,dc=example,dc=com")
    (version 3.0; acl "Client authorization for database links"; allow (all)
    userdn = "ldap:///uid=*,c=us,ou=people,dc=example,dc=com";)
```

   This ACI allows clients that have a UID in **c=us,ou=people,dc=example,dc=com** on Server 1 to perform any type of operation on the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** suffix tree on server three. If there are users on Server 2 under a different suffix that will require additional rights on server three, it may be necessary to add additional client ACIs on Server 2.

2.4.6.3. Configuring Server Three

1. Create an administrative user on server three for Server 2 to use for proxy authorization:

```
dn: cn=server2 proxy admin,cn=config
```

```
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: server2 proxy admin
sn: server2 proxy admin
userPassword: secret
description: Entry for use by database links
```

2. Then add the same local proxy authorization ACI to server three as on Server 2. Add the following proxy authorization ACI to the **l=Zanzibar,ou=people,dc=example,dc=com** entry:

```
aci: (targetattr = "*")(version 3.0; acl "Proxied authorization
    for database links"; allow (proxy) userdn = "ldap:///cn=server2
    proxy admin,cn=config";)
```

   This ACI gives the Server 2 proxy admin read-only access to the data contained on the remote server, server three, within the **l=Zanzibar,ou=people,dc=example,dc=com** subtree only.

3. Create a local client ACI on the **l=Zanzibar,ou=people,dc=example,dc=com** subtree that corresponds to the original client application. Use the same ACI as the one created for the client on Server 2:

```
aci: (targetattr ="*")(target="l=Zanzibar,c=africa,ou=people,dc=example,dc=com")
    (version 3.0; acl "Client authentication for database link users"; allow (all)
    userdn = "ldap:///uid=*,c=us,ou=people,dc=example,dc=com";)
```

The cascading chaining configuration is now set up. This cascading configuration allows a user to bind to Server 1 and modify information in the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** branch on server three. Depending on your security needs, it may be necessary to provide more detailed access control.

2.5. USING REFERRALS

Referrals tell client applications which server to contact for a specific piece of information. This redirection occurs when a client application requests a directory entry that does not exist on the local server or when a database has been taken off-line for maintenance. This section contains the following information about referrals:

- Section 2.5.1, “Starting the Server in Referral Mode”
- Section 2.5.2, “Setting Default Referrals”
- Section 2.5.3, “Creating Smart Referrals”
- Section 2.5.4, “Creating Suffix Referrals”

For conceptual information on how to use referrals in the directory, see the *Directory Server Deployment Guide*.

2.5.1. Starting the Server in Referral Mode

Referrals are used to redirect client applications to another server while the current server is unavailable or when the client requests information that is not held on the current server. For example, starting Directory Server in referral mode while there are configuration changes being made to the Directory Server will refer all clients to another supplier while that server is unavailable. Starting the Directory Server in referral mode is done with the **refer** command.

Run **nsslapd** with the **refer** option.

```
/usr/sbin/ns-slapd refer -D /etc/dirsrv/slapd-instance_name [-p port] -r referral_url
```

- **/etc/dirsrv/slapd-*instance_name*** is the directory where the Directory Server configuration files are. This is the default location on Red Hat Enterprise Linux 6 (64-bit).
- *port* is the optional port number of the Directory Server to start in referral mode.
- *referral_url* is the referral returned to clients. The format of an LDAP URL is covered in Appendix C, *LDAP URLs*.

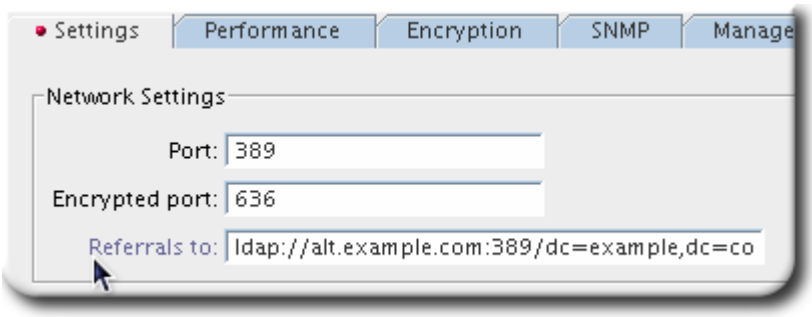
2.5.2. Setting Default Referrals

Default referrals are returned to client applications that submit operations on a DN not contained within any of the suffixes maintained by the directory. The following procedures describes setting a default referral for the directory using the console and the command-line utilities.

2.5.2.1. Setting a Default Referral Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Select the top entry in the navigation tree in the left pane.

3. Select the **Settings** tab in the right pane.
4. Enter an LDAP URL for the referral.



Enter multiple referral URLs separated by spaces and in quotes:

```
"ldap://dir1.example.com:389/dc=example,dc=com" "ldap://dir2.example.com/"
```

For more information about LDAP URLs, see Appendix C, *LDAP URLs*.

2.5.2.2. Setting a Default Referral from the Command Line

ldapmodify can add a default referral to the **cn=config** entry in the directory's configuration file. For example, to add a new default referral from one Directory Server, **dir1.example.com**, to a server named **dir2.example.com**, add a new line to the **cn=config** entry.

1. Run the **ldapmodify** utility and add the default referral to the **dir2.example.com** server:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-referral
nsslapd-referral: ldap://dir2.example.com/
```

After adding the default referral to the **cn=config** entry of the directory, the directory will return the default referral in response to requests made by client applications. The Directory Server does not need to be restarted.

2.5.3. Creating Smart Referrals

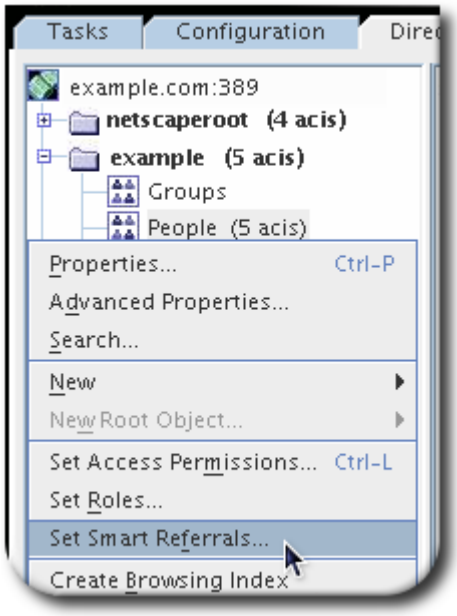
Smart referrals map a directory entry or directory tree to a specific LDAP URL. Using smart referrals, client applications can be referred to a specific server or a specific entry on a specific server.

For example, a client application requests the directory entry **uid=jdoe,ou=people,dc=example,dc=com**. A smart referral is returned to the client that points to the entry **cn=john doe,o=people,l=europe,dc=example,dc=com** on the server **directory.europe.example.com**.

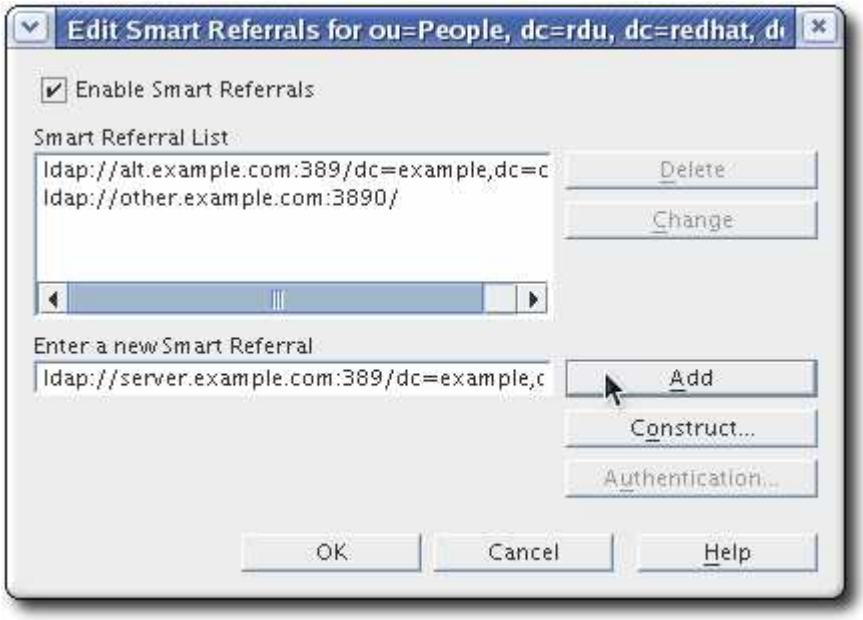
The way the directory uses smart referrals conforms to the standard specified in RFC 2251 section 4.1.11. The RFC can be downloaded at http://www.ietf.org/rfc/rfc2251.txt.

2.5.3.1. Creating Smart Referrals Using the Directory Server Console

1. In the Directory Server Console, select the **Directory** tab.
2. Browse through the tree in the left navigation pane, and select the entry for which to add the referral.
3. Right-click the entry, and select **Set Smart Referrals**.



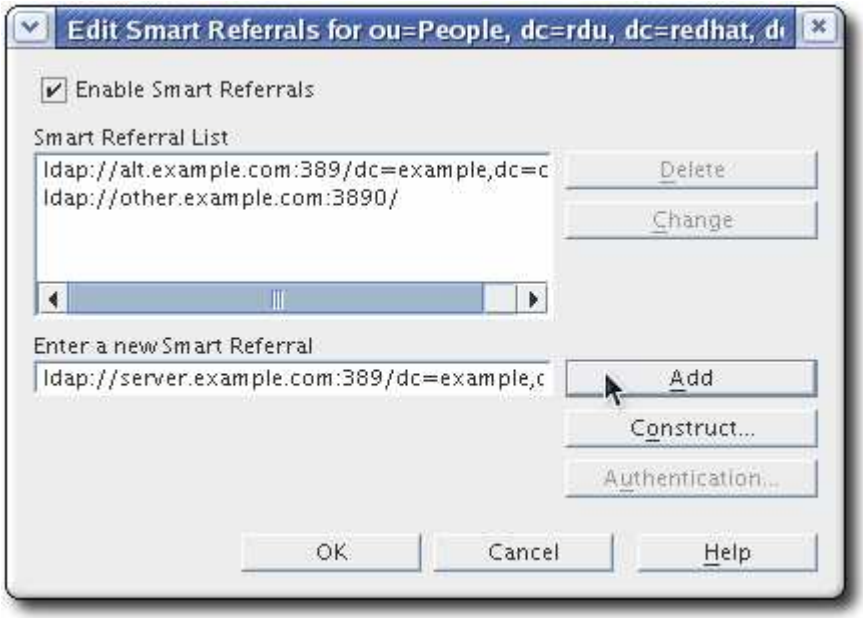
4. Select the **Enable Smart Referral** check box. (Unchecking the option removes all smart referrals from the entry and deletes the **referral** object class from the entry.)



5. In the **Enter a new Smart Referral** field, enter a referral in the LDAP URL format, and then click **Add**. The LDAP URL must be in the following format:

```
ldap://server:port/[optional_dn]
```

server can be the host name, IPv4 address, or IPv6 address for the server. *optional_dn* is the explicit DN for the server to return to the requesting client application.

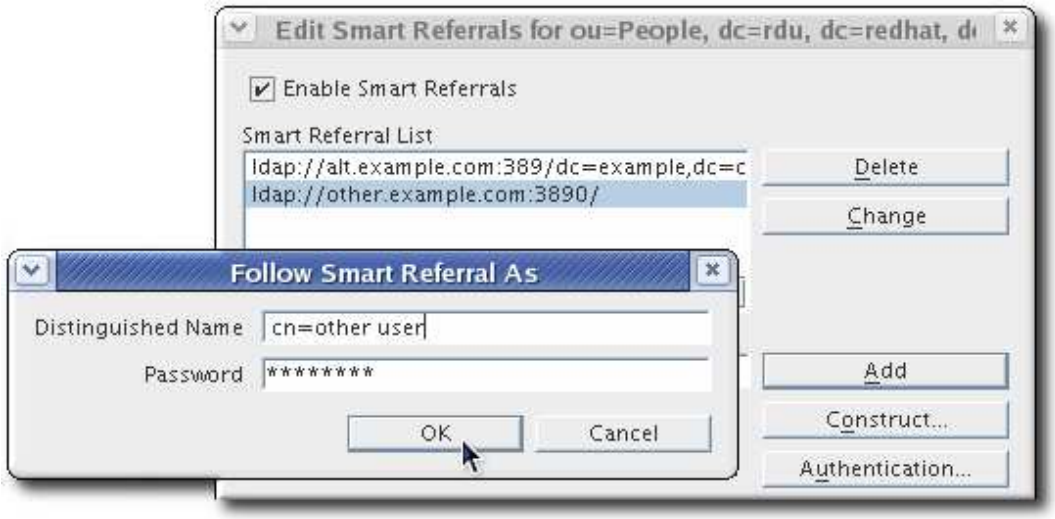


Construct opens a wizard to direct the process of adding a referral.

The **Smart Referral List** lists the referrals currently in place for the selected entry. The entire list of referrals is returned to client applications in response to a request with the **Return Referrals for All Operations** or **Return Referrals for Update Operations** options in the **Suffix Settings** tab, which is available under the **Configuration** tab.

To modify the list, click **Edit** to edit the selected referral or **Delete** to delete the selected referral.

6. To set the referral to use different authentication credentials, click **Authentication**, and specify the appropriate DN and password. This authentication remains valid only until the Console is closed; then it is reset to the same authentication used to log into the Console.



2.5.3.2. Creating Smart Referrals from the Command Line

Use the **ldapmodify** command-line utility to create smart referrals from the command line.

To create a smart referral, create the relevant directory entry, and add the **referral** object class. This object class allows a single attribute, ***ref***. The ***ref*** attribute must contain an LDAP URL.

For example, add the following to return a smart referral for an existing entry, **uid=jdoe**:

```
dn: uid=jdoe,ou=people,dc=example,dc=com
objectclass: referral
ref: ldap://directory.europe.example.com/cn=john%20doe,ou=people,l=europe,dc=example,dc=com
```

NOTE

Any information after a space in an LDAP URL is ignored by the server. For this reason, use **%20** instead of spaces in any LDAP URL used as a referral.

To add the entry **uid=jdoe,ou=people,dc=example,dc=com** with a referral to **directory.europe.example.com**, include the following in the LDIF file before importing:

```
dn: uid=jdoe,ou=people,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: referral
cn: john doe
sn: doe
uid: jdoe
ref: ldap://directory.europe.example.com/cn=john%20doe,ou=people,l=europe,dc=example,dc=com
```

Use the **-M** option with **ldapmodify** when there is already a referral in the DN path. For more information on smart referrals, see the *Directory Server Deployment Guide*.

2.5.4. Creating Suffix Referrals

The following procedure describes creating a referral in a *suffix*. This means that the suffix processes operations using a referral rather than a database or database link.

WARNING

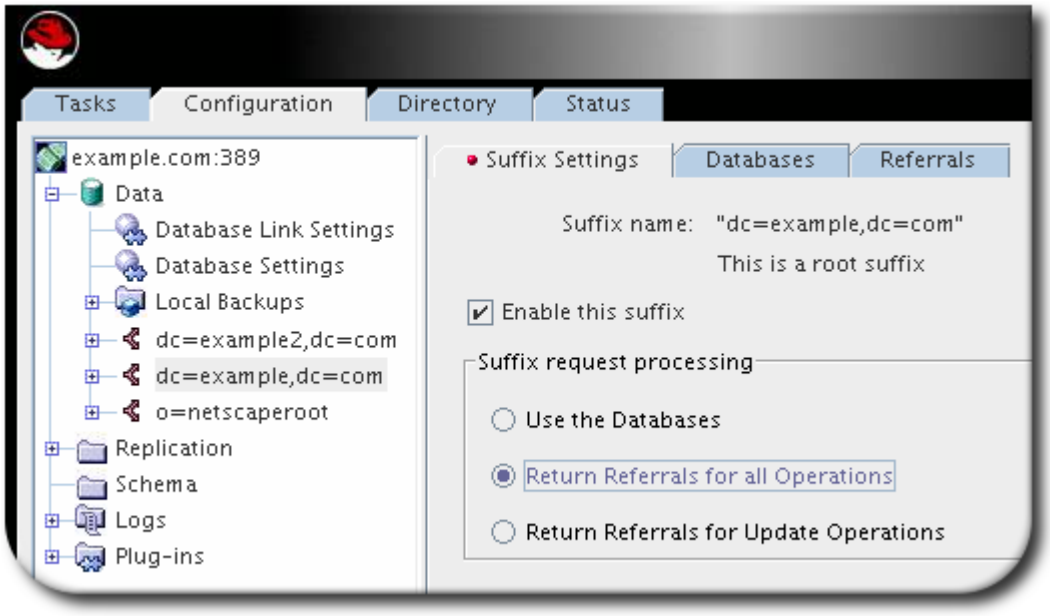
When a suffix is configured to return referrals, the ACIs contained by the database associated with the suffix are ignored.

2.5.4.1. Creating Suffix Referrals Using the Console

Referrals can be used to point a client application temporarily to a different server. For example, adding a referral to a suffix so that the suffix points to a different server allows the database associated with the suffix is taken off-line for maintenance without affecting the users of the Directory Server database.

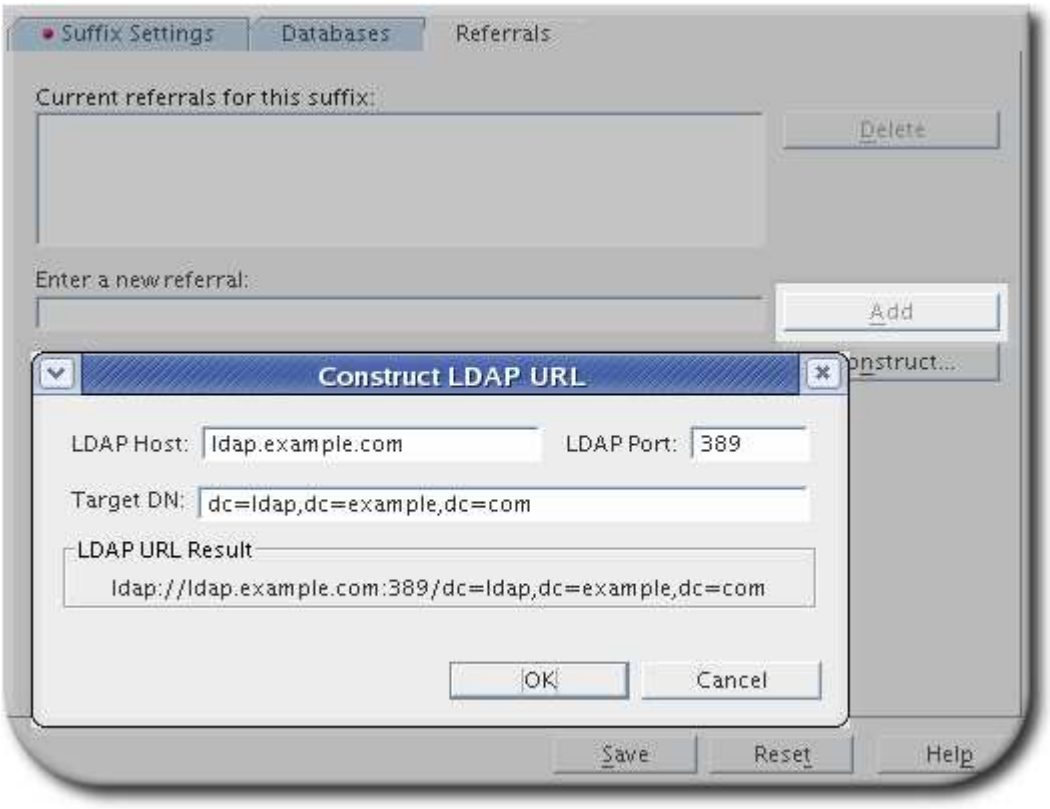
To set referrals in a suffix:

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left pane, select the suffix for which to add a referral.
3. Click the **Suffix Settings** tab, and select the **Return Referrals for ... Operations** radio button.



Selecting **Return Referrals for Update Operations** means that the directory redirects only update and write requests to a read-only database. For example, there may be a local copy of directory data, and that data should be available for searches but not for updates, so it is replicated across several servers. Enabling referrals for that Directory Server only for update requests means that when a client asks to update an entry, the client is referred to the server that owns the data, where the modification request can proceed.

4. Click the **Referrals** tab. Enter an LDAP URL in the [1] in the **Enter a new referral** field, or click **Construct** to create an LDAP URL.



5. Click **Add** to add the referral to the list.

   You can enter multiple referrals. The directory returns the entire list of referrals in response to requests from client applications.

2.5.4.2. Creating Suffix Referrals from the Command Line

Add a suffix referral to the root or sub suffix entry in the directory configuration file under the **cn=mapping tree,cn=config** branch.

Run **ldapmodify** and add a suffix referral to the **ou=people,dc=example,dc=com** root suffix:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ou=people,dc=example,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: extensibleObject
objectclass: nsMappingTree
nsslapd-state: referral
nsslapd-referral: ldap://zanzibar.com/
```

The ***nsslapd-state*** attribute is set to **referral**, meaning that a referral is returned for requests made to this suffix. The ***nsslapd-referral*** attribute contains the LDAP URL of the referral returned by the suffix, in this case a referral to the **zanzibar.com** server.

The ***nsslapd-state*** attribute can also be set to **referral on update**. This means that the database is used for all operations except update requests. When a client application makes an update request to a suffix set to **referral on update**, the client receives a referral.

For more information about the suffix configuration attributes, see Table 2.1, “Suffix Attributes”.

[] Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It has the form **ldap://***server:port*/, where *server* can be the host name, IPv4 address, or IPv6 address.

[1] Appendix C, *LDAP URLs* has more information about the structure of LDAP URLs.

CHAPTER 3. CREATING DIRECTORY ENTRIES

This chapter discusses how to use the Directory Server Console and the **ldapmodify** and **ldapdelete** command-line utilities to modify the contents of your directory.

Entries stored in Active Directory can be added to the Directory Server through Windows Sync; see Chapter 12, *Synchronizing Red Hat Directory Server with Microsoft Active Directory* for more information on adding or modifying synchronized entries through Windows User Sync.

3.1. MANAGING ENTRIES FROM THE DIRECTORY CONSOLE

You can use the **Directory** tab and the **Property Editor** on the Directory Server Console to add, modify, or delete entries individually.

To add several entries simultaneously, use the command-line utilities described in Section 3.2, "Managing Entries from the Command Line".

- Section 3.1.1, "Creating a Root Entry"
- Section 3.1.2, "Creating Directory Entries"
- Section 3.1.3, "Modifying Directory Entries"
- Section 3.1.4, "Deleting Directory Entries"

NOTE

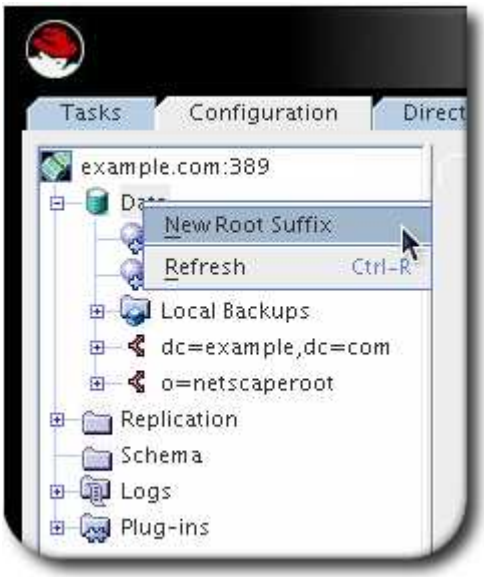
You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for your directory, see Chapter 13, *Managing Access Control*.

3.1.1. Creating a Root Entry

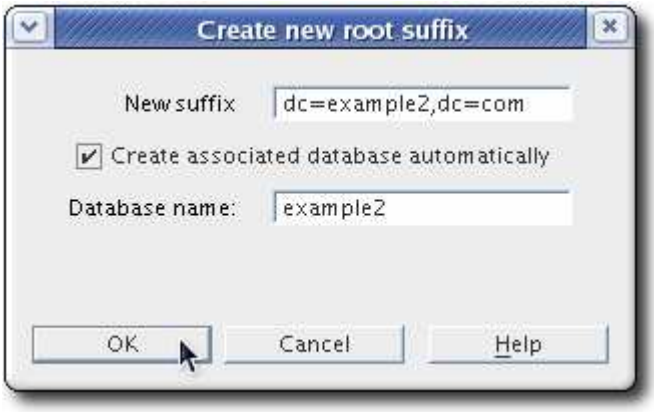
Each time a new database is created, it is associated with the suffix that will be stored in the database. The directory entry representing that suffix is not automatically created.

To create a root entry for a database:

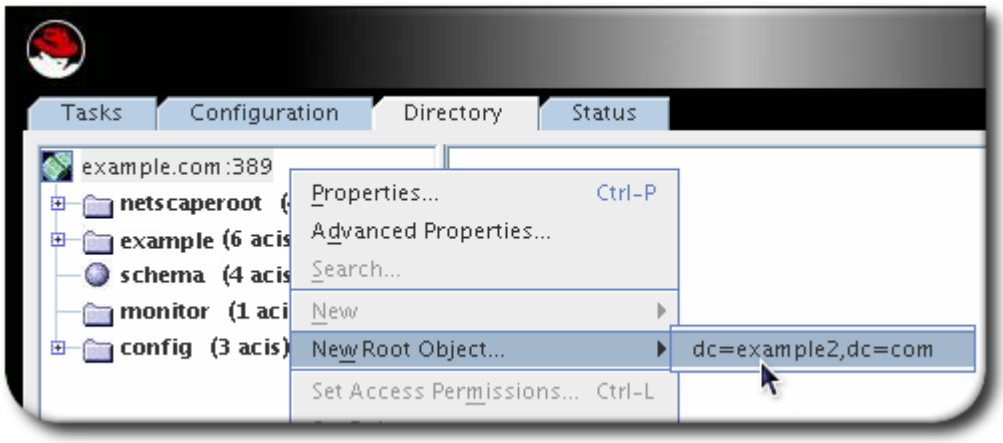
1. In the Directory Server Console, select the **Configuration** tab.
2. Right-click on the **Data** entry in the left menu, and select **New Root Suffix** from the menu.



3. Fill in the new suffix and database information.

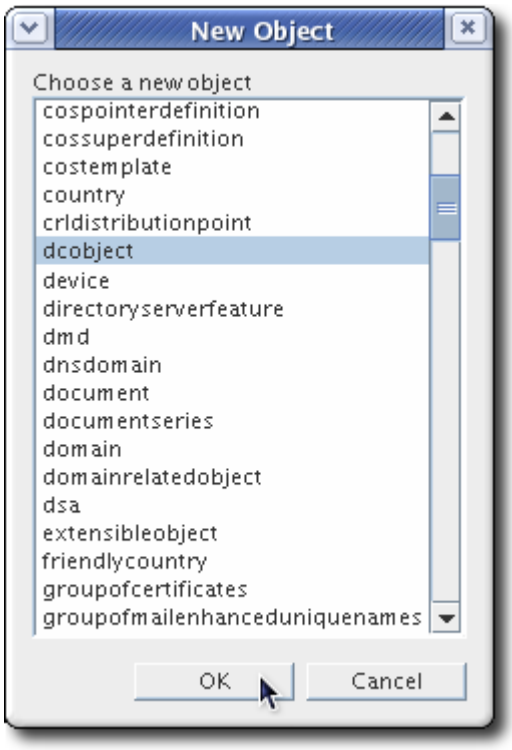


4. In the **Directory** tab, right-click the top object representing the Directory Server, and choose **New Root Object**.



The secondary menu under **New Root Object** displays the new suffixes without a corresponding directory entry. Choose the suffix corresponding to the entry to create.

5. In the **New Object** window, select the object class corresponding to the new entry.



The object class must contain the attribute used to name the suffix. For example, if the entry corresponds to the suffix **ou=people,dc=example,dc=com**, then choose the **organizationalUnit** object class or another object class that allows the **ou** attribute.

6. Click **OK** in the New Object window.

The **Property Editor** for the new entry opens. You can either accept the current values by clicking **OK** or modify the entry, as explained in Section 3.1.3, "Modifying Directory Entries" .

3.1.2. Creating Directory Entries

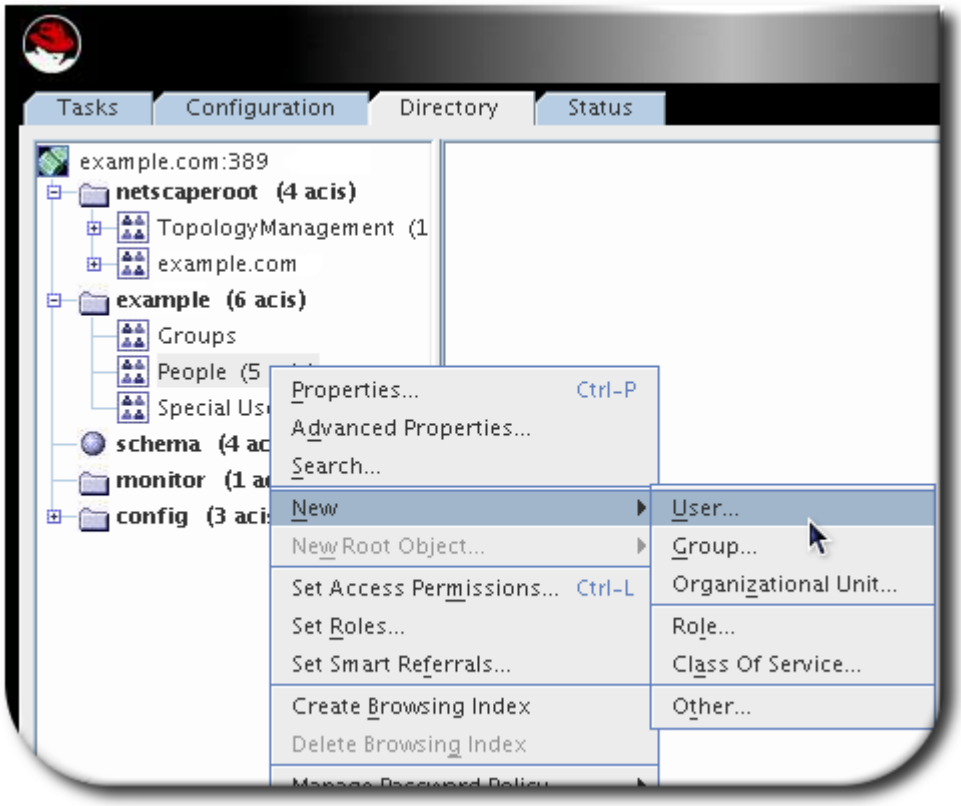
Directory Server Console offers predefined templates, with preset forms, for new directory entries. Table 3.1, "Entry Templates and Corresponding Object Classes" shows what type of object class is used for each template.

Table 3.1. Entry Templates and Corresponding Object Classes

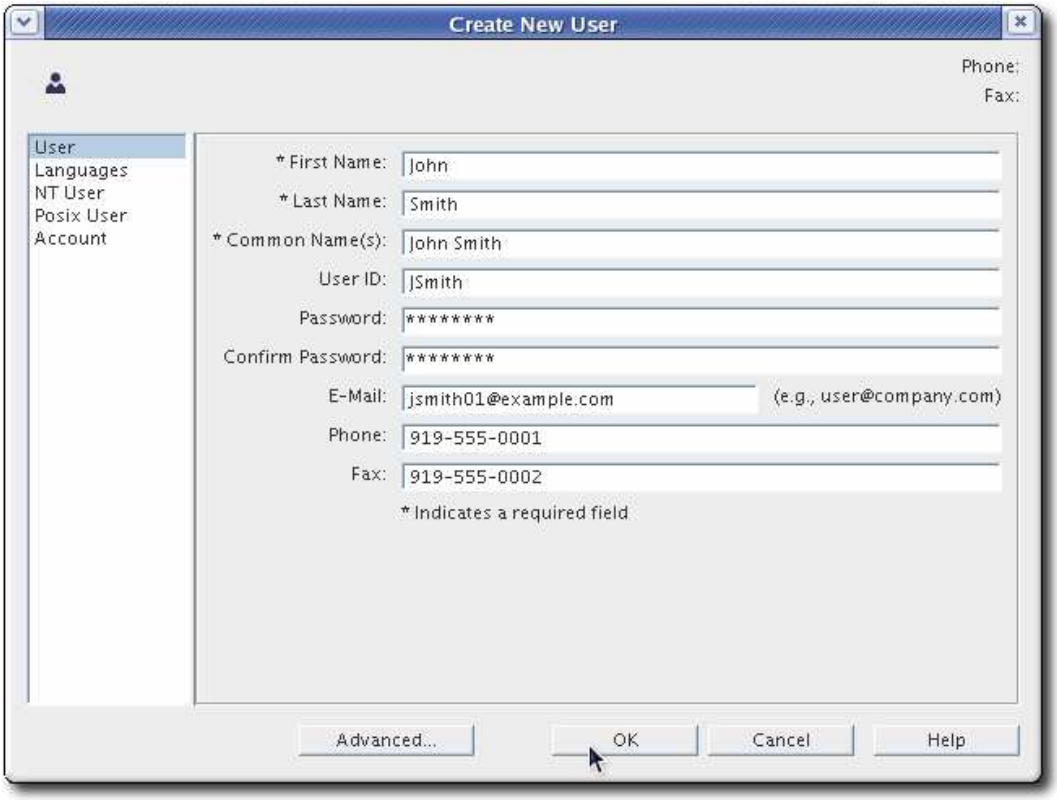
Template	Object Class
User	inetOrgPerson
Group	groupOfUniqueNames
Organizational Unit	organizationalUnit
Role	nsRoleDefinition
Class of Service	cosSuperDefinition

Another type, **Other** allows any kind of entry to be created by allowing users to select the specific object classes and attributes to apply.

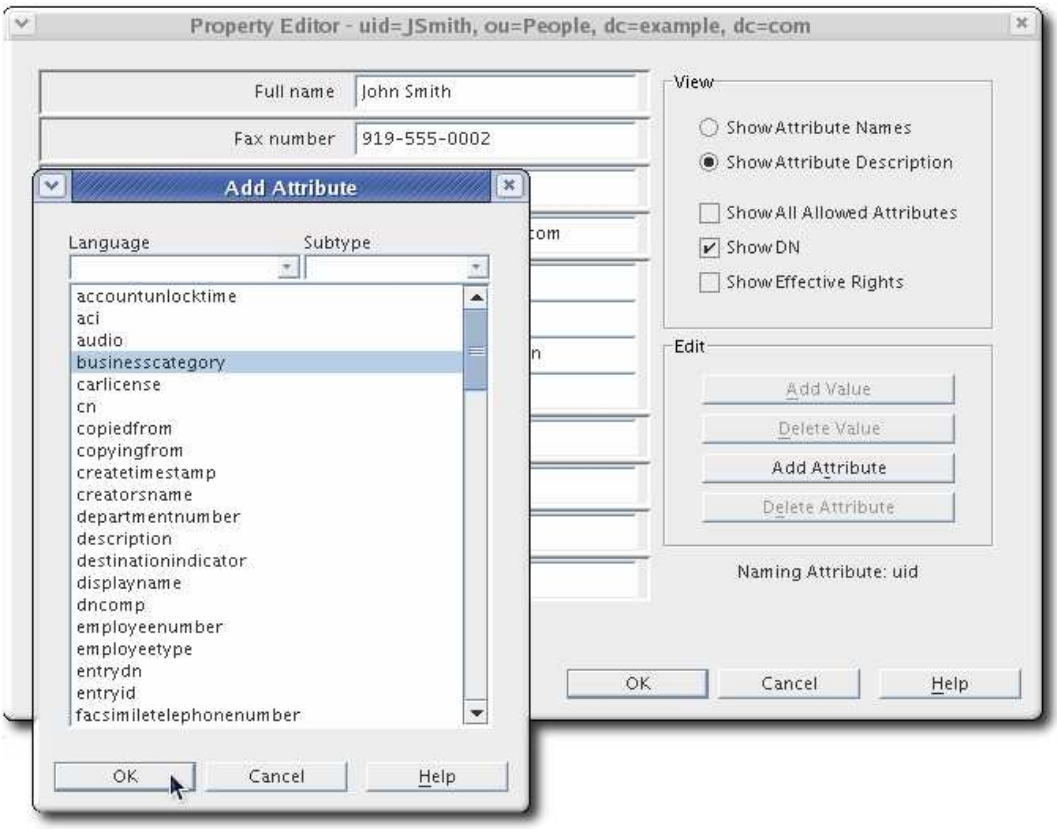
1. In the Directory Server Console, select the **Directory** tab.
2. In the left pane, right-click the main entry to add the new entry, and select the type of entry: **User**, **Group**, **Organizational Unit**, **Role**, **Class of Service**, or **Other**.



3. If the new entry type was **Other**, then a list of object classes opens. Select an object class from the list to define the new entry.
4. Supply a value for all the listed attributes. Required attributes are marked with an asterisk (*).

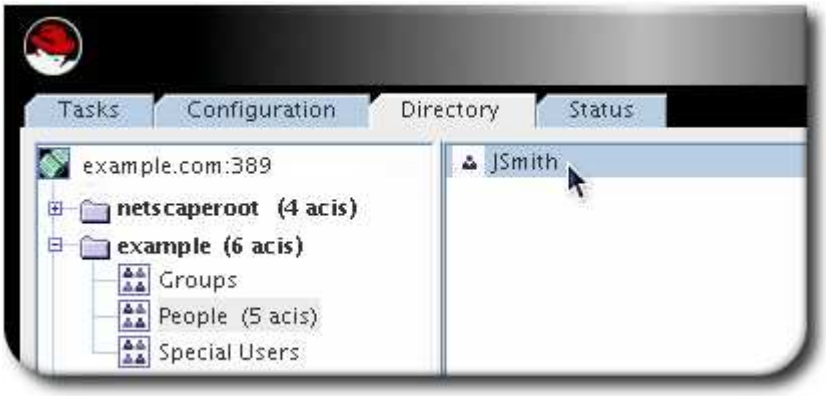


5. To display the full list of attributes available for the object class (entry type), click the **Advanced** button.



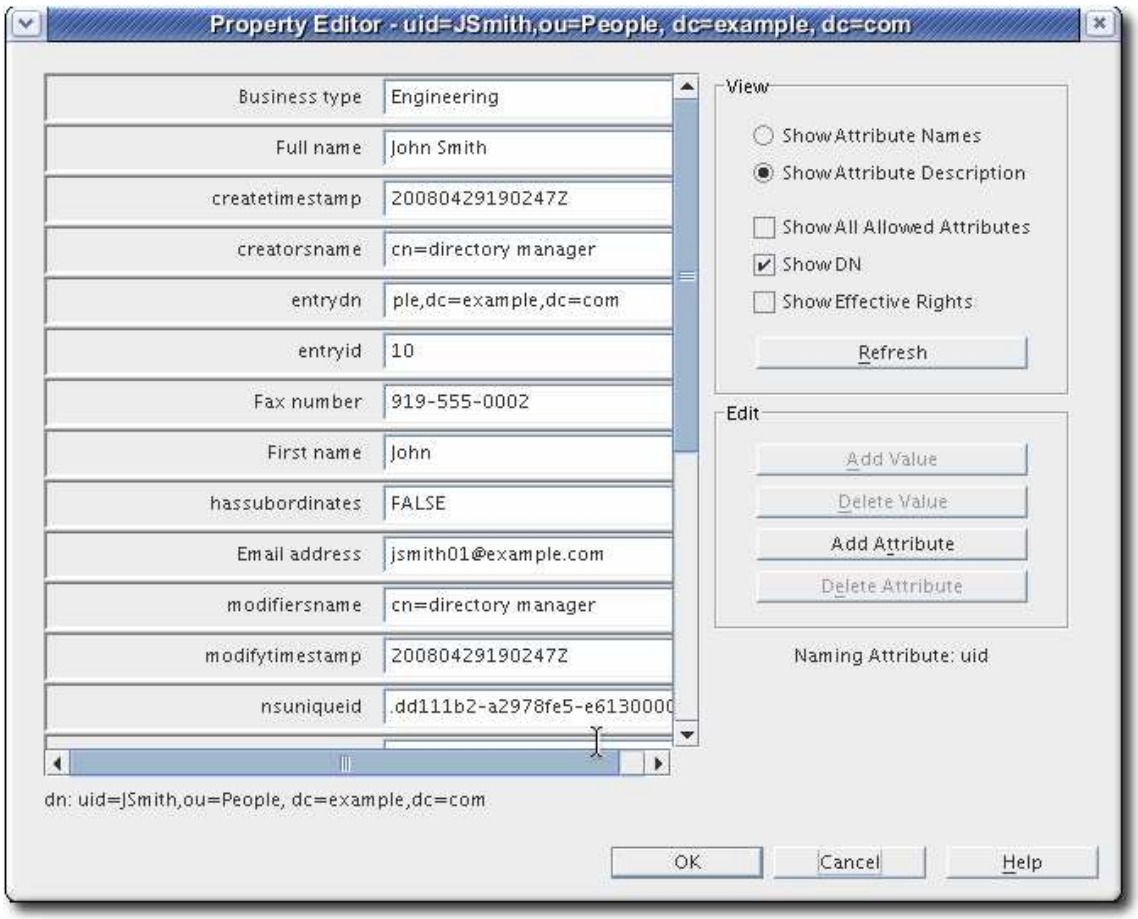
In the **Property Editor**, select any additional attributes, and fill in the attribute values.

6. Click **OK** to save the entry. The new entry is listed in the right pane.



3.1.3. Modifying Directory Entries

Modifying directory entries in Directory Server Console uses a dialog window called the **Property Editor**. The **Property Editor** contains the list of object classes and attributes belonging to an entry and can be used to edit the object classes and attributes belonging to that entry by adding and removing object classes, attributes and attribute values, and attribute subtypes.



The **Property Editor** can be opened in several ways:

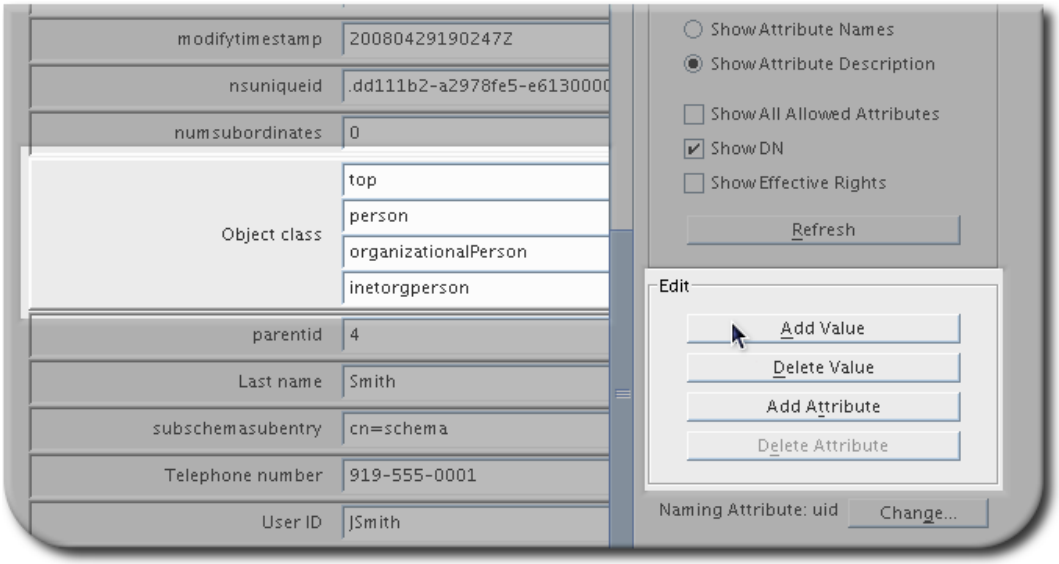
- From the **Directory** tab, by right-clicking an entry, and selecting **Advanced Properties** from the pop-up menu.
- From the **Directory** tab, by double-clicking an entry and clicking the **Advanced** button
- From the **Create...** new entry forms, by clicking the **Advanced** button
- From the **New Object** window, by clicking **OK**

3.1.3.1. Adding or Removing an Object Class to an Entry

To add an object class to an entry:

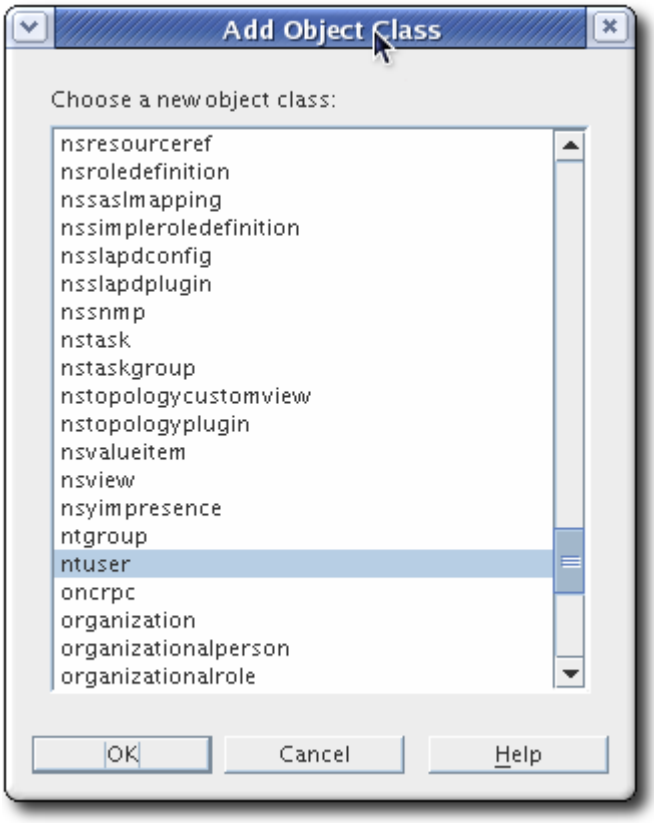
1. In the **Directory** tab of the Directory Server Console, right-click the entry to modify, and select **Advanced** from the pop-up menu.

2. Select the object class field, and click **Add Value**.



The **Add Object Class** window opens. It shows a list of object classes that can be added to the entry.

3. Select the object class to add, and click **OK**.



To remove an object class from an entry, click the text box for the object class to remove, and then click **Delete Value**.

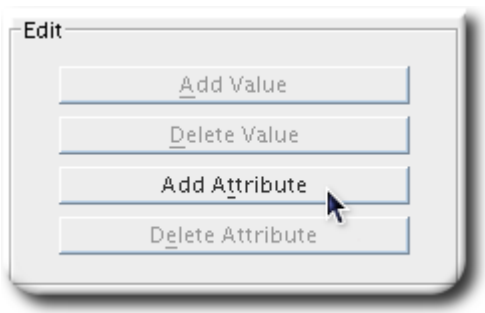
3.1.3.2. Adding an Attribute to an Entry

Before you can add an attribute to an entry, the entry must contain an object class that either requires or allows the attribute. See Section 3.1.3.1, "Adding or Removing an Object Class to an Entry" and Chapter 8, *Managing the Directory Schema* for more information.

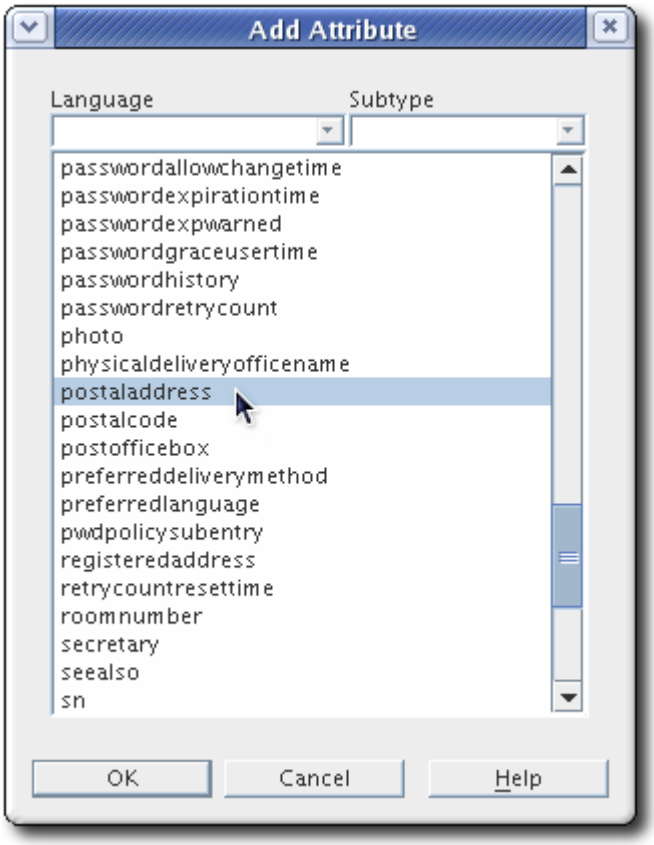
To add an attribute to an entry:

1. In the **Directory** tab of the Directory Server Console, right-click the entry to modify, and select **Advanced** from the pop-up menu.

2. Click **Add Attribute**.



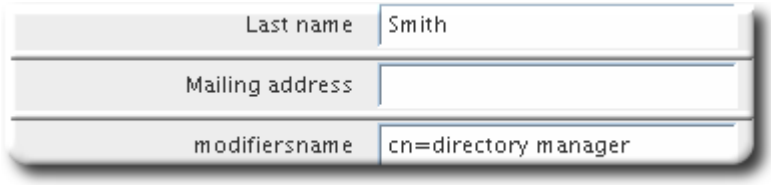
3. Select the attribute to add from the list, and click **OK**.



NOTE

If the attribute you want to add is not listed, add the object class containing the attribute first, then add the attribute. See Section 3.1.3.1, "Adding or Removing an Object Class to an Entry" for instructions on adding an object class. If you do not know which object class contains the attribute you need, look up the attribute in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*, which lists the object classes which use that attribute.

4. Type in the value for the new attribute in the field to the right of the attribute name.



To remove the attribute and all its values from the entry, select **Delete Attribute** from the **Edit** menu.

3.1.3.3. Adding Very Large Attributes

The configuration attribute ***nsslapd-maxbersize*** sets the maximum size limit for LDAP requests. The default configuration of Directory Server sets this attribute at 2 megabytes. LDAP add or modify operations will fail when attempting to add very large attributes that result in a request that is larger than 2 megabytes.

To add very large attributes, first change the setting for the ***nsslapd-maxbersize*** configuration attribute to a value larger than the largest LDAP request you will make.

When determining the value to set, consider *all* elements of the LDAP add and modify operations used to add the attributes, not just the single attribute. There are a number of different factors to consider, including the following:

- The size of each attribute name in the request
- The size of the values of each of the attributes in the request
- The size of the DN in the request

- Some overhead, usually 10 kilobytes

One common issue that requires increasing the ***nsslapd-maxbersize*** setting is using attributes which hold CRL values, such as ***certificateRevocationList***, ***authorityRevocationList***, and ***deltaRevocationList***.

Note that this limit is additionally applied to replication processes.

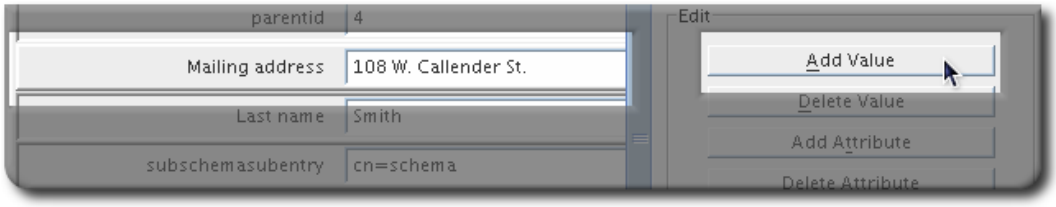
For further information about the ***nsslapd-maxbersize*** attribute and a workaround in replication scenarios, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

3.1.3.4. Adding Attribute Values

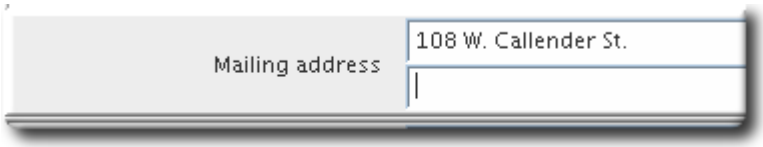
Multi-valued attributes allow multiple value for one attribute to be added to an entry.

1. In the **Directory** tab of the Directory Server Console, right-click the entry to modify, and select **Advanced** from the pop-up menu.

2. Select the attribute to which to add a value, and then click **Add Value**.



3. Type in the new attribute value.



To remove an attribute value from an entry, click the text box of the attribute value to remove, and click **Delete Value**.

3.1.3.5. Adding an Attribute Subtype

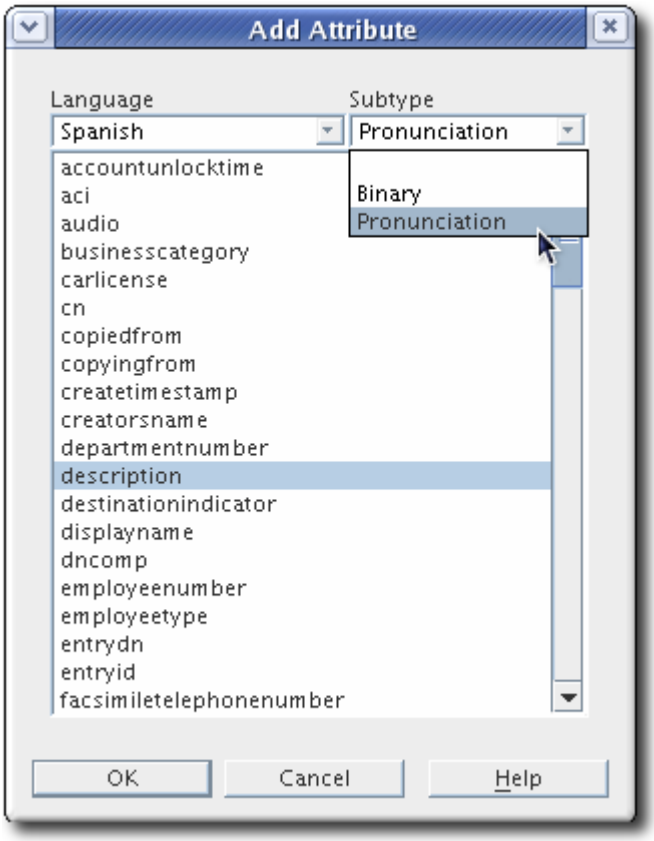
A subtype allows the same entry value to be represented in different ways, such as providing a foreign-characterset version. There three different kinds of subtypes to attributes which can be added to an entry: language, binary, and pronunciation.

To add a subtype to an entry:

1. In the **Directory** tab of the Directory Server Console, right-click the entry to modify, and select **Properties** from the pop-up menu.

2. Click **Add Attribute**, and select the attribute to add from the list.

3. Add a language subtype by selecting a value from the **Language** drop-down list. Add either a binary or pronunciation subtype by selecting a value from the **Subtype** drop-down list.



Language Subtype

Sometimes a user's name can be more accurately represented in characters of a language other than the default language. For example, a user, Noriko, has a name in Japanese and prefers that her name be represented by Japanese characters when possible. You can select Japanese as a language subtype for the **givenname** attribute so that other users can search for her name in Japanese as well as English. For example:

```
givenname;lang-ja
```

To specify a language subtype for an attribute, add the subtype to the attribute name as follows:

```
attribute;lang-subtype:attribute value
```

attribute is the attribute being added to the entry and *subtype* is the two character abbreviation for the language. The supported language subtypes are listed in Table D.2, "Supported Language Subtypes".

Only one language subtype can be added per attribute *instance* in an entry. To assign multiple language subtypes, add another attribute instance to the entry, and then assign the new language subtype. For example, the following is illegal:

```
cn;lang-ja;lang-en-GB:value
```

Instead, use:

```
cn;lang-ja:ja-value
cn;lang-en-GB:value
```

Binary Subtype

Assigning the binary subtype to an attribute indicates that the attribute value is binary, such as user certificates (**usercertificate;binary**).

Although you can store binary data within an attribute that does not contain the **binary** subtype (for example, **jpegphoto**), the **binary** subtype indicates to clients that multiple variants of the attribute type may exist.

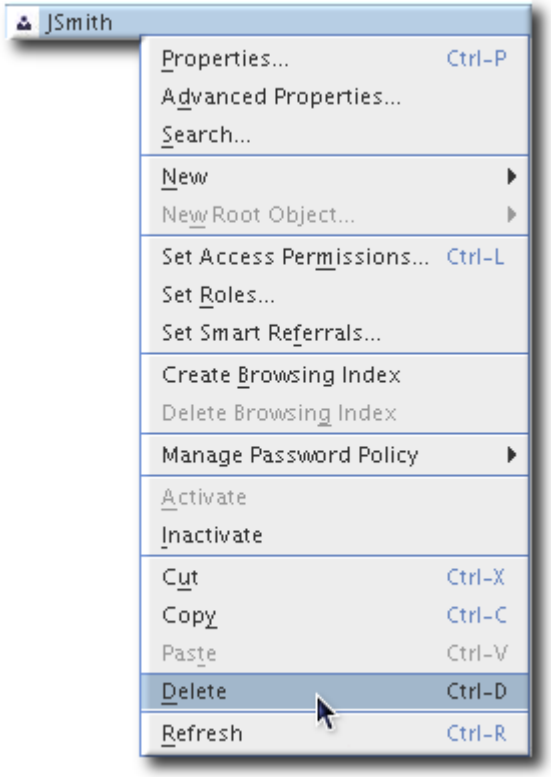
Pronunciation Subtype

Assigning the pronunciation subtype to an attribute indicates that the attribute value is a phonetic representation. The subtype is added to the attribute name as *attribute***;phonetic**. This subtype is commonly used in combination with a language subtype for languages that have more than one alphabet, where one is a phonetic representation.

This subtype is useful with attributes that are expected to contain user names, such as **cn** or **givenname**. For example, **givenname;lang-ja;phonetic** indicates that the attribute value is the phonetic version of the user's Japanese name.

3.1.4. Deleting Directory Entries

1. In the Directory Server Console, select the **Directory** tab.
2. Right-click the entry to delete, and select **Delete** from the right-click menu.



WARNING

The server deletes the entry or entries immediately. There is no way to undo the delete operation.

3.2. MANAGING ENTRIES FROM THE COMMAND LINE

The command-line utilities allow you to manipulate the contents of your directory. They can be useful to write scripts to perform bulk management of the directory or to test the Directory Server. For example, you might want to ensure that it returns the expected information after you have made changes to access control information.

With command-line utilities, information can be provided directly from the command line or through an LDIF input file.

- Section 3.2.1, “Providing Input from the Command Line”
- Section 3.2.2, “Creating a Root Entry from the Command Line”
- Section 3.2.3, “Adding Entries Using LDIF”
- Section 3.2.4, “Adding and Modifying Entries Using ldapmodify”
- Section 3.2.5, “Deleting Entries Using ldapdelete”
- Section 3.2.6, “Using Special Characters”

NOTE

You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for the directory, see Chapter 13, *Managing Access Control*.

3.2.1. Providing Input from the Command Line

When you provide input to the **ldapmodify** and **ldapdelete** utilities directly from the command line, you must use LDIF statements. For detailed information on LDIF statements, see Section 3.3, "Using LDIF Update Statements to Create or Modify Entries".

The **ldapmodify** and **ldapdelete** utilities read the statements that you enter in exactly the same way as if they were read from a file. When all of the input has been entered, enter the character that the shell recognizes as the end of file (EOF) escape sequence. The utility then begins operations based on the supplied inputs.

While the EOF escape sequence depends on the type of machine, the EOF escape sequence almost always control-D (**^D**).

For example, to input some LDIF update statements to **ldapmodify**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: cn=Barry Nixon,ou=people,dc=example,dc=com
 changetype: modify
 delete: telephonenumber
 -
 add: manager
 manager: cn=Harry Cruise,ou=people,dc=example,dc=com
 ^D
```

When adding an entry from the command line or from LDIF, make sure that an entry representing a subtree is created before new entries are created under that branch. For example, to place an entry in a **People** subtree, create an entry representing that subtree before creating entries within the subtree. For example:

```
dn: dc=example,dc=com
dn: ou=People,dc=example,dc=com
...People subtree entries. ...
dn: ou=Group,dc=example,dc=com
...Group subtree entries. ...
```

3.2.2. Creating a Root Entry from the Command Line

The **ldapmodify** command-line utility can be used to create a new root entry in a database. For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: Suffix_Name
changetype: add
objectclass: newobjectclass
```

The DN corresponds to the DN of the root or sub-suffix contained by the database. The *newobjectclass* value depends upon the type of object class you are adding to the database. You may need to specify additional required attributes depending on the type of root object being added.

NOTE

You can use **ldapmodify** to add root objects only if you have one database per suffix. If you create a suffix that is stored in several databases, you must use the **ldif2db** utility with the **-noption** parameter to specify the database that will hold the new entries. For information, see Section 4.1.6, "Importing from the Command Line".

3.2.3. Adding Entries Using LDIF

You can use an LDIF file to add multiple entries or to import an entire database. To add entries using an LDIF file and the Directory Server Console:

1. Define the entries in an LDIF file.

   LDIF files are described in Appendix B, *LDAP Data Interchange Format*.

2. Import the LDIF file from the Directory Server Console.

   See Section 4.1.4, "Importing a Database from the Console" for information about LDIF file formats. When you import the LDIF file, select **Append to database** in the **Import** dialog box so that the server will only import entries that do not currently exist in the directory.

You can also add entries described in an LDIF file from the command line using the **ldapmodify** command with the **-f** option.

3.2.4. Adding and Modifying Entries Using ldapmodify

The **ldapmodify** command can add and modify entries in an existing Directory Server database. The **ldapmodify** command opens a connection to the specified server using the supplied distinguished name and password and modifies the entries based on LDIF update statements contained in a specified file. Because **ldapmodify** uses LDIF update statements, **ldapmodify** can do everything that **ldapdelete** can do.

Consider the following when using **ldapmodify**:

- If the server detects an attribute or object class in the entry that is not known to the server, then the modify operation will fail when it reaches the erroneous entry. All entries that were processed before the error was encountered will be successfully added or modified. If you run **ldapmodify** with the **-c** option (do not stop on errors), all correct entries processed after the erroneous entry will be successfully added or modified.
- If a required attribute is not present, the modify operation fails. This happens even if the offending object class or attribute is not being modified.

NOTE

To create the root entry a database suffix (such as **dc=example,dc=com**) using **ldapmodify**, you must bind to the directory as the Directory Manager.

3.2.4.1. Adding Entries Using ldapmodify

Typically, to add the entries using **ldapmodify**, pass the **-a** option to indicate an add operation and the LDIF file to use which contains the new entry information (and, optionally, the bind credentials and any connection information). For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x -f new.ldif
```

The entries to be created are specified in the file **new.ldif**. (In this example, the LDIF statements in the **new.ldif** file do not specify a change type. They follow the format defined in Section B.1, "About the LDIF File Format".)

If the new entry is not passed in a given LDIF file, then the **ldapmodify** utility waits for the DN of the new entry and then each object class and attribute for the entry, each on a new line in LDIF format. When the last attribute is entered, hit enter twice to submit the new entry.

Table 3.2, "ldapmodify Parameters Used for Adding Entries" describes the **ldapmodify** parameters used in the example.

Table 3.2. ldapmodify Parameters Used for Adding Entries

Parameter Name	Description
-a	Specifies that the modify operation will add new entries to the directory.
-D	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
-w	Specifies the password associated with the distinguished name specified in the **-D** parameter.
-h	Specifies the name of the host on which the server is running.

Parameter Name	Description
-p	Specifies the port number that the server uses.
-f	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from **stdin**. For information on supplying LDIF update statements from the command line, see Section 3.2.1, "Providing Input from the Command Line".

3.2.4.2. Modifying Entries Using ldapmodify

Typically, to edit entries using **ldapmodify**, specify the DN and password to bind to the Directory Server, the port and host of the Directory Server, and the LDIF file to use, as when adding entries with **ldapmodify**. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x -f modify_statements
```

The entries to modify are specified in the file **modify_statements**. Before the entries can be modified, you must first create the **modify_statements** file with the appropriate LDIF update statements; LDIF update statements are described in Section 3.3, "Using LDIF Update Statements to Create or Modify Entries" .

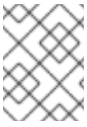
Table 3.3, "ldapmodify Parameters Used for Modifying Entries" describes the **ldapmodify** parameters used in the example.

Table 3.3. ldapmodify Parameters Used for Modifying Entries

Parameter Name	Description
-D	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
-w	Specifies the password associated with the distinguished name specified in the **-D** parameter.
-h	Specifies the name of the host on which the server is running.
-p	Specifies the port number that the server uses.
-f	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from **stdin**. For information on supplying LDIF update statements from the command line, see Section 3.2.1, "Providing Input from the Command Line".
-x	Disables SASL to allow a simple bind to the server.

3.2.5. Deleting Entries Using ldapdelete

The **ldapdelete** command-line utility opens a connection to the specified server using the provided distinguished name and password and deletes the specified entry or entries.



NOTE

You can only delete entries at the end of a branch. You cannot delete entries that are branch points in the directory tree.

For example, of the following three entries, only the last two entries can be deleted.

```
ou=People,dc=example,dc=com
cn=Paula Simon,ou=People,dc=example,dc=com
cn=Jerry O'Connor,ou=People,dc=example,dc=com
```

The entry that identifies the **People** subtree can be deleted only if there are not any entries below it. To delete **ou=People,dc=example,dc=com**, you must first delete Paula Simon and Jerry O'Connor's entries and all other entries in that subtree.

Like **ldapmodify**, running **ldapdelete** requires the DN and password to bind to the Directory Server, the port and host of the Directory Server, and the DNs of the entries to delete. For example:

```
ldapdelete -D "cn=directory manager" -w secret -p 389 -h server.example.com -x "cn=Robert
Jenkins,ou=People,dc=example,dc=com" "cn=Lisa Jangles,ou=People,dc=example,dc=com"
```

The DNs of the entries to delete (**cn=Robert Jenkins,ou=People,dc=example,dc=com** and **cn=Lisa Jangles,ou=People,dc=example,dc=com**) are appended to the end of the delete command.

Table 3.4, "ldapdelete Parameters Used for Deleting Entries" describes the **ldapdelete** parameters used in the example:

Table 3.4. ldapdelete Parameters Used for Deleting Entries

Parameter Name	Description
-D	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
-w	Specifies the password associated with the distinguished name specified in the **-D** parameter.
-h	Specifies the name of the host on which the server is running.
-p	Specifies the port number that the server uses.
-x	Disables SASL to allow a simple bind to the server.

3.2.6. Using Special Characters

When using the Directory Server command-line client tools, you may need to specify values that contain characters that have special meaning to the command-line interpreter, such as space ( ), asterisk (*), or backslash (\). When this situation occurs, enclose the value in quotation marks (""). For example:

```
-D "cn=Barbara Jensen,ou=Product Development,dc=example,dc=com"
```

Depending on the command-line utility, use either single or double quotation marks; see your operating system documentation for more information.

Additionally, if a DN contains commas, you must escape the commas with a backslash (\). For example:

```
-D "cn=Patricia Fuentes,ou=people,o=example.com Bolivia\,S.A."
```

To delete user Patricia Fuentes from the **example.com Bolivia, S.A.** tree, use the following command:

```
ldapdelete -D "cn=directory manager" -w secret -p 389 -h server.example.com -x "cn=Patricia
Fuentes,ou=People,o=example.com Bolivia\,S.A."
```

3.3. USING LDIF UPDATE STATEMENTS TO CREATE OR MODIFY ENTRIES

LDIF update statements define how **ldapmodify** changes the directory entry. In general, LDIF update statements contain the following information:

- The DN of the entry to be modified.

- A changetype that defines how a specific entry is to be modified (**add**, **delete**, **modify**, **modrdn**).
- A series of attributes and their changed values.

A change type is required unless **ldapmodify** is run with the **-a** parameter. If you specify the **-a** parameter, then an add operation (**changetype: add**) is assumed. However, any other change type overrides the **-a** parameter.

If you specify a modify operation (**changetype: modify**), a change operation is required that indicates how the entry should be changed.

If you specify **changetype: modrdn**, change operations are required that specify how the relative distinguished name (RDN) is to be modified. A distinguished name's RDN is the left-most value in the DN. For example, the distinguished name **uid=ssarette,dc=example,dc=com** has an RDN of **uid=ssarette**.

The general format of LDIF update statements is as follows:

```
dn: distinguished_name
changetype: changetype_identifier
change_operation_identifier: list_of_attributes
change_operation_identifier: list_of_attributes
```

A dash (-) must be used to denote the end of a change operation if subsequent change operations are specified. For example, the following statement adds the telephone number and manager attributes to the entry:

```
dn: cn=Lisa Jangles,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: (408) 555-2468
-
add: manager
manager: cn=Harry Cruise,ou=People,dc=example,dc=com
```

In addition, the line continuation operator is a single space. Therefore, the following two statements are identical:

```
dn: cn=Lisa Jangles,ou=People,dc=example,dc=com

dn: cn=Lisa Jangles,
 ou=People,
 dc=example,dc=com
```

The following sections describe the change types in detail.

3.3.1. Adding an Entry Using LDIF

changetype: add adds an entry to the directory. When you add an entry, make sure to create an entry representing a branch point before you try to create new entries under that branch. That is, to place an entry in a **People** and a **Groups** subtree, then create the branch point for those subtrees before creating entries within the subtrees. For example:

```
dn: dc=example,dc=com
changetype: add
objectclass: top
objectclass: organization
o: example.com

dn: ou=People,dc=example,dc=com
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: People
ou: Marketing

dn: cn=Pete Minsky,ou=People,dc=example,dc=com
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Pete Minsky
givenName: Pete
```

```
sn: Minsky
ou: People
ou: Marketing
uid: pminsky

dn: cn=Sue Jacobs,ou=People,dc=example,dc=com
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Sue Jacobs
givenName: Sue
sn: Jacobs
ou: People
ou: Marketing
uid: sjacobs

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

dn: cn=Administrators,ou=Groups,dc=example,dc=com
changetype: add
objectclass: top
objectclass: groupOfNames
member: cn=Sue Jacobs,ou=People,dc=example,dc=com
member: cn=Pete Minsky,ou=People,dc=example,dc=com
cn: Administrators

dn: ou=example.com Bolivia\, S.A.,dc=example,dc=com
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: example.com Bolivia\, S.A.

dn: cn=Carla Flores,ou=example.com Bolivia\,S.A.,dc=example,dc=com
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Carla Flores
givenName: Carla
sn: Flores
ou: example.com Bolivia\, S.A.
uid: cflores
```

3.3.2. Renaming an Entry Using LDIF

changetype: modrdn changes an entry's distinguished name. Most commonly, it changes the *relative DN*, or RDN, the left-most element in the distinguished name. The RDN for **cn=Barry Nixon,ou=People,dc=example,dc=com** is **cn=Barry Nixon**, and the RDN for **ou=People,dc=example,dc=com** is **ou=People**.

The following command renames Sue Jacobs to Susan Jacobs:

```
dn: cn=Sue Jacobs,ou=Marketing,dc=example,dc=com
changetype: modrdn
newrdn: cn=Susan Jacobs
deleteoldrdn: 0
```

Because **deleteoldrdn** is **0**, this example retains the existing RDN as a value in the new entry. The resulting entry would therefore have a common name (**cn**) attribute set to both Sue Jacobs and Susan Jacobs, in addition to all the other attributes included in the original entry. However, using the following command causes the server to delete **cn=Sue Jacobs**, so that only **cn=Susan Jacobs** remains in the entry:

```
dn: cn=Sue Jacobs,ou=Marketing,dc=example,dc=com
changetype: modrdn
newrdn: cn=Susan Jacobs
```

```
deleteoldrdn: 1
```

While it is most common for the **modrdn** operation to be used in a direct name change operation, it can be used to change other parts of the DN, which is both a rename operation *and* a move operation. **modrdn** can update the name of all leaf entries if the name of a subtree entry changes, or a single leaf can be moved to a new parent by changing its DN. This is covered in more detail in Section 3.4, "Renaming and Moving Entries".

3.3.3. Modifying an Entry Using LDIF

changetype: modify can add, replace, or remove attributes or attribute values in an entry. When you specify **changetype: modify**, you must also provide a change operation to indicate how the entry is to be modified. Change operations can be as follows:

- **add:** *attribute*

  Adds the specified attribute or attribute value. If the attribute type does not currently exist for the entry, then the attribute and its corresponding value are created. If the attribute type already exists for the entry, then the specified attribute value is added to the existing value. If the particular attribute value already exists for the entry, then the operation fails, and the server returns an error.

- **replace:** *attribute*

  The specified values are used to entirely replace the attribute's values. If the attribute does not already exist, it is created. If no replacement value is specified for the attribute, the attribute is deleted.

- **delete:** *attribute*

  The specified attribute is deleted. If more than one value of an attribute exists for the entry, then all values of the attribute are deleted in the entry. To delete just one of many attribute values, specify the attribute and associated value on the line following the delete change operation.

This section contains the following topics:

- Section 3.3.3.1, "Adding Attributes to Existing Entries Using LDIF"
- Section 3.3.3.2, "Changing an Attribute Value Using LDIF"
- Section 3.3.3.3, "Deleting All Values of an Attribute Using LDIF"
- Section 3.3.3.4, "Deleting a Specific Attribute Value Using LDIF"

3.3.3.1. Adding Attributes to Existing Entries Using LDIF

Using **changetype: modify** with the add operation cam add an attribute and an attribute value to an entry. For example, the following LDIF update statement adds a telephone number to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
```

The following example adds two telephone numbers to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
telephonenumber: 555-6789
```

The following example adds two **telephonenumber** attributes and a **manager** attribute to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
telephonenumber: 555-6789
-
add: manager
manager: cn=Sally Nixon,ou=People,dc=example,dc=com
```

The following example adds a **jpeg** photograph to the directory. In order to add this attribute to the directory, use the **-b** parameter, which indicates that **ldapmodify** should read the referenced file for binary values if the attribute value begins with a slash:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: jpegphoto
jpegphoto: /path/to/photo
```

You can also add a **jpeg** photograph to the directory using the following standard LDIF notation:

```
jpegphoto: < file:/path/to/photo
```

Using the standard notation means that the **-b** parameter does not need to be used with **ldapmodify**. However, you must add **version:1** to the beginning of the LDIF file or with LDIF update statements. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
 version: 1
 dn: cn=Barney Fife,ou=People,dc=example,dc=com
 changetype: modify
 add: userCertificate
 userCertificate;binary:< file: BarneysCert
```

NOTE

Standard LDIF notation can *only* be used with the **ldapmodify** command, not with other command-line utilities.

3.3.3.2. Changing an Attribute Value Using LDIF

changetype: modify with the replace operation changes all values of an attribute in an entry. For example, the following LDIF update statement changes Barney's manager from Sally Nixon to Wally Hensford:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
replace: manager
manager: cn=Wally Hensford,ou=People,dc=example,dc=com
```

If the entry has multiple instances of the attribute, then to change one of the attribute values, you must delete the attribute value first and then add the replacement value. For example, this entry has two telephone numbers:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-1212
telephonenumber: 555-6789
```

To change the telephone number **555-1212** to **555-4321**, use the following LDIF update statement:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
telephonenumber: 555-1212
-
add: telephonenumber
telephonenumber: 555-4321
```

The entry is now as follows:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-6789
telephonenumber: 555-4321
```

3.3.3.3. Deleting All Values of an Attribute Using LDIF

changetype: modify with the delete operation deletes an attribute from an entry. If the entry has more than one instance of the attribute, you must indicate which of the attributes to delete.

For example, the following LDIF update statement deletes all instances of the ***telephonenumber*** attribute from the entry, regardless of how many times it appears in the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
```

To delete just a specific instance of the ***telephonenumber*** attribute, simply delete that specific attribute value, as described in the next section.

3.3.3.4. Deleting a Specific Attribute Value Using LDIF

Running **changetype: modify** with the delete operation can delete a single value for an attribute value from an entry, as well as deleting all instances of the attribute. For example, consider the following entry:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-1212
telephonenumber: 555-6789
```

To delete the **555-1212** telephone number from this entry, use the following LDIF update statement:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
telephonenumber: 555-1212
```

Barney's entry then becomes:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-6789
```

3.3.4. Deleting an Entry Using LDIF

changetype: delete is the change type which deletes an entire entry from the directory.

NOTE

You can only delete leaf entries. Therefore, when you delete an entry, make sure that no other entries exist under that entry in the directory tree. That is, you cannot delete an organizational unit entry unless you have first deleted all the entries that belong to the organizational unit.

For example, of the following three entries, only the last two entries can be deleted:

```
ou=People,dc=example,dc=com
cn=Paula Simon,ou=People,dc=example,dc=com
cn=Jerry O'Connor,ou=People,dc=example,dc=com
```

The entry that identifies the **People** subtree can be deleted only if no other entries exist below it.

The following LDIF update statements can be used to delete person entries:

```
dn: cn=Pete Minsky,ou=People,dc=example,dc=com
changetype: delete
dn: cn=Sue Jacobs,ou=People,dc=example,dc=com
changetype: delete
```

WARNING

Do not delete the suffix **o=NetscapeRoot**. The Admin Server uses this suffix to store information about installed Directory Servers. Deleting this suffix could force you to reinstall the Directory Server.

3.3.5. Modifying an Entry in an Internationalized Directory

If the attribute values in the directory are associated with languages other than English, the attribute values are associated with language tags. When using the **ldapmodify** command-line utility to modify an attribute that has an associated language tag, you must match the value and language tag exactly or the modify operation will fail.

For example, to modify an attribute value that has a language tag of **lang-fr**, include **lang-fr** in the modify operation, as follows:

```
dn: bjensen,dc=example,dc=com
changetype: modify
replace: homePostalAddress;lang-fr
homePostalAddress;lang-fr: 34 rue de Seine
```

3.4. RENAMING AND MOVING ENTRIES

Most rename operations are done on a leaf entry. However, some types of rename operations result in changes to the directory tree itself. These operations – subtree renames and new superior operations – are possible because of the way that entry IDs are stored in Directory Server indexes.

3.4.1. About Renaming Entries

Every DN is comprised of a long chain of elements that build on one another. Every entry has its own name, and then each of its children combine their name (the element on the far left) with the parent name. This pattern orients entries within the directory tree.

Example 3.1. Building Entry DNs

```
                                    dc=example,dc=com  =>  root suffix
                                ou=People,dc=example,dc=com  =>  org unit
                            st=California,ou=People,dc=example,dc=com  =>  state/province
                        l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  city
                    ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  org unit
uid=jsmith,ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  leaf entry
```

Rename, or **modrdn**, operations result in changes to this tree.

3.4.1.1. Types of Rename Operations

When the naming attribute of an entry, the leftmost element of the DN, is changed, this is a *modrdn operation*. That's a special kind of modify operation because, in a sense, it moves the entry within the directory tree. For leaf entries (entries with no children), **modrdn** operations are lateral moves; the entry has the same parent, just a new name.

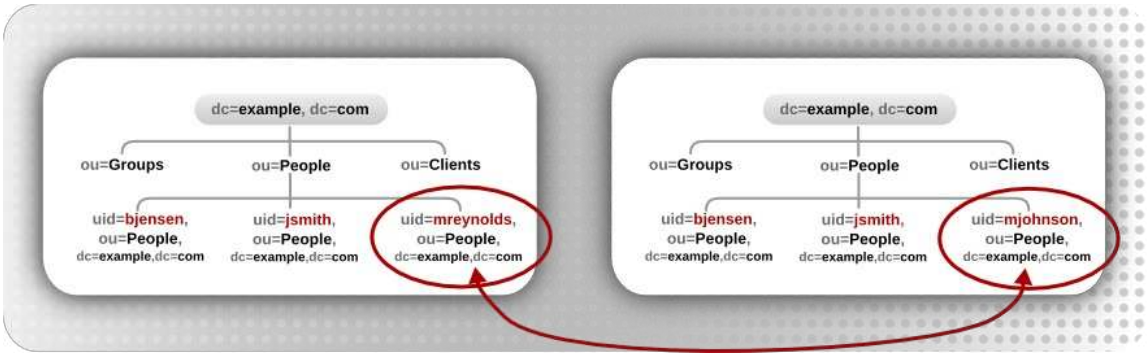


Figure 3.1. modrdn Operations for a Leaf Entry

For subtree entries, the **modrdn** operation not only renames the subtree entry itself, but also changes the DN components of all of the children entries *beneath* the subtree.

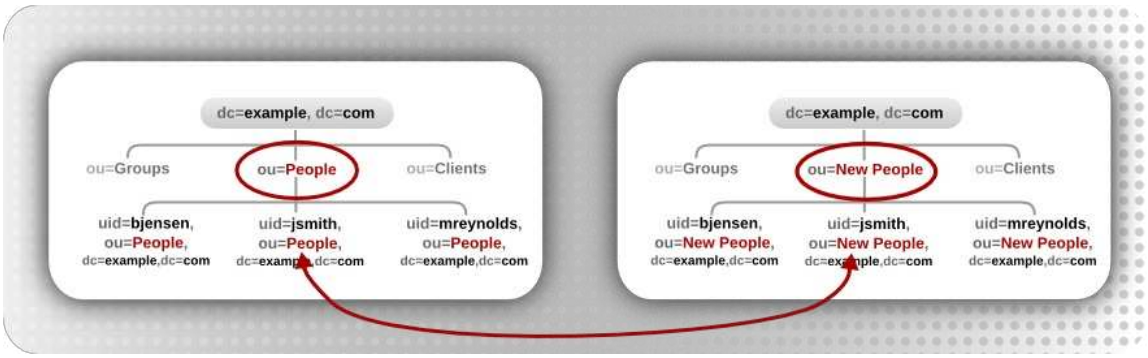


Figure 3.2. modrdn Operations for a Subtree Entry

IMPORTANT

Subtree **modrdn** operations also move and rename all of the child entries beneath the subtree entry. For large subtrees, this can be a time- and resource-intensive process. Plan the naming structure of your directory tree hierarchy so that it will not require frequent subtree rename operations.

A similar action to renaming a subtree is moving an entry from one subtree to another. This is an expanded type of **modrdn** operation, which simultaneously renames the entry (even if it is the same name) and sets a ***newSuperior*** attribute which moves the entry from one parent to another.

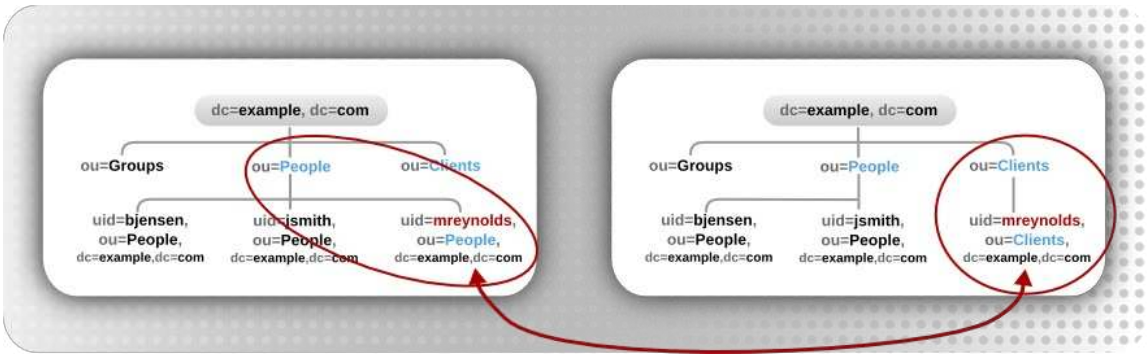


Figure 3.3. modrdn Operations to a New Parent Entry

3.4.1.2. The Relationships Between Entry Keys in entryrdn.db4

Both new superior and subtree rename operations are possible because of how entries are stored in the **entryrdn.db4** index. Each entry is identified by its own key (a *self-link*) and then a subkey which identifies its parent (the *parent link*) and any children. This has a format that lays out the directory tree hierarchy by treating parents and children as attribute to an entry, and every entry is describes by a unique ID and its RDN, rather than the full DN.

These relationships are visible in the **dbscan** output for the **entryrdn.db4** index.

```
numeric_id:RDN => self link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
```

```
P#:RDN => parent link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
C#:RDN => child link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
```

For example, the **ou=people** subtree has a parent of **dc=example,dc=com** and a child of **uid=jsmith**.

```
4:ou=people
  ID: 4; RDN: "ou=People"; NRDN: "ou=people"
P4:ou=people
  ID: 1; RDN: "dc=example,dc=com"; NRDN: "dc=example,dc=com"
C4:ou=people
  ID: 10; RDN: "uid=jsmith"; NRDN: "uid=jsmith"
```

NOTE

Version 8.2 and older of Directory Server used the **entrydn.db4** index, which stored entry keys according to their full DN. To upgrade to **entryrdn.db4**, run the **dn2rdn** tool.

The defined keys for parent and child entries makes it possible for the Directory Server to perform operations that move entries in the directory tree.

- Add operations look at the parent given in the modify statement and check for any existing leaf entries; if none exist, the new entry is added.
- For delete operations, the server first checks the self link (the full entry key), and then deletes both the self link and then the child link in the parent entry. For example, if you deleted **ou=people,dc=example,dc=com**, the server deletes the **ou=people** key (the self-link) and then deletes the child subkey from the **dc=example,dc=com** entry key.
- For new superior operations, the server changes the parent link in the entry key, removes the child link from the old parent, and adds a new child link to the new superior key.
- For subtree rename operations, the server changes the self link in the subtree entry key to update its RDN, then tracks any child entry keys (based on the child links) and updates the parent link RDN.

3.4.1.3. Considerations for Renaming Entries

There are some things to keep in mind when performing rename operations:

- You cannot rename the root suffix.
- Subtree rename operations have minimal effect on replication. Replication agreements are applied to an entire database, not a subtree within the database, so a subtree rename operation does not require re-configuring a replication agreement. All of the name changes after a subtree rename operation are replicated as normal.
- Renaming a subtree **may** require any synchronization agreements to be re-configured. Sync agreements are set at the suffix or subtree level, so renaming a subtree may break synchronization.
- Renaming a subtree **requires** that any subtree-level ACIs set for the subtree be re-configured manually, as well as any entry-level ACIs set for child entries of the subtree.
- You can rename a subtree with children, but you cannot delete a subtree with children.
- Trying to change the component of a subtree, like moving from ***ou*** to ***dc***, may fail with a schema violation. For example, the **organizationalUnit** object class requires the ***ou*** attribute. If that attribute is removed as part of renaming the subtree, then the operation will fail.
- Any ***memberOf*** attributes managed by the MemberOf Plug-in will **not** be updated after a subtree-level rename operation. The MemberOf Plug-in does not check to see if *parent* entries change in order to initiate updates. If a subtree is renamed which contains either groups or group members, then launch a **cn=memberof task** task or use the **fixup-memberof.pl** command to force the MemberOf Plug-in to make the changes.

  See Section 6.1.4.6, “Synchronizing memberOf Values” to see how to clean up ***memberOf*** attribute references.

3.4.2. Renaming an Entry or Subtree

To rename an entry or subtree, use the **modrdn** changetype LDIF statement and use the ***newrdn*** attribute to indicate that the naming attribute is being changed and ***deleteoldrdn*** to set whether to retain the old RDN (0) or delete it (any non-zero integer). For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: ou=Engineering,ou=People,dc=example,dc=com
changetype: modrdn
newrdn: ou=Development
deleteoldrdn: 1
```

If ***deleteoldrdn*** is set to any non-zero integer, positive or negative, then the old RDN is removed.

LDIF statements for **modrdn** operations are described in Section 3.3.2, "Renaming an Entry Using LDIF".

3.4.3. Moving an Entry to a New Parent

Changing the parent entry requires setting two attributes:

- ***newrdn*** to set the RDN of the moved entry. This is required even if the RDN remains the same.
- ***newSuperior*** to give the DN of the new parent entry.

For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,l=Boston,ou=Engineering,ou=People,dc=example,dc=com
changetype: modrdn
newrdn: uid=jsmith
deleteoldrdn: 1
newSuperior: l=Mountain View,ou=Engineering,ou=People,dc=example,dc=com
```

3.4.4. Disabling Subtree Rename Operations

By default, renaming subtrees is allowed. However, this can be disabled by resetting the ***nsslapd-subtree-rename-switch*** parameter to **off**:

NOTE

When a new Directory Server 9.0 instance or an upgraded instance has subtree renames enabled, then the RDNs of the entries are mapped in the **entryrdn.db4** database. Instances with subtree rename disabled store their entry information in **entrydn.db4**. When disabling subtree renames, then the server needs to switch from using one database to the other, which is done by exporting and re-importing the directory entries. This is described in Section 3.4.1.2, "The Relationships Between Entry Keys in entryrdn.db4".

1. Export the directory to LDIF.

   ```
   /etc/dirsrv/slapd-instance_name/db2ldif -n database1 -a /var/lib/dirsrv/slapd-instance_name/ldif/tree.ldif
   ```

   More export options are covered in Section 4.2, "Exporting Data".

2. Turn off the ***nsslapd-subtree-rename-switch*** parameter.

   ```
   [root@server ~]# ldapmodify -D "cn=directory manager" -W -x
   dn: cn=config,cn=ldbm database,cn=plugins,cn=config
   changetype: modify
   replace: nsslapd-subtree-rename-switch
   nsslapd-subtree-rename-switch: off
   ```

3. Import the LDIF file. This will populate the new **entrydn.db4** file.

   ```
   /etc/dirsrv/slapd-instance_name/ldif2db -D "cn=Directory Manager" -w secret -i
   /var/lib/dirsrv/slapd-instance_name/ldif/tree.ldif -n database1
   ```

   More import options are covered in Section 4.1, "Importing Data".

3.4.5. Setting the DN Cache Size

A separate cache is used specifically to store DN information. As with the entry cache, the DN cache uses memory (RAM) used to store directory entries information in the internal representation. This is configured in the ***nsslapd-dncachememsize***, and setting a large cache memory size improves the performance of rename operations.

The cache itself stores only the entry ID and the entry DN, so the table is relatively small when compared to the entry cache.

For the best move performance, make the entry cache large enough to contain all entries in the database.

This value can be modified from the command line by editing the ***nsslapd-dncachememsize*** attribute value for the LDBM plug-in instance for the database. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-dncachememsize
nsslapd-dncachememsize: 20971520
```

3.5. TRACKING MODIFICATIONS TO DIRECTORY ENTRIES

It can be useful to track when changes are made to entries. There are two aspects of entry modifications that the Directory Server tracks:

- Using change sequence numbers to track changes to the database. This is similar to change sequence numbers used in replication and synchronization. Every normal directory operation triggers a sequence number.
- Assigning creation and modification information. These attributes record the names of the user who created and most recently modified an entry, as well as the timestamps of when it was created and modified.

NOTE

The entry USN, modify time and name, and create time and name are all operational attributes and are not returned in a regular **ldapsearch**. For details on running a search for operational attributes, see Section 10.5.7, “Searching for Operational Attributes”.

3.5.1. Tracking Modifications to the Database through Update Sequence Numbers

The USN Plug-in provides a way for LDAP clients to know that something – anything – in the database has changed.

3.5.1.1. An Overview of the Entry Sequence Numbers

When the USN Plug-in is enabled, update sequence numbers (USNs) are sequential numbers that are assigned to an entry whenever a write operation is performed against the entry. (Write operations include add, modify, modrdn, and delete operations. Internal database operations, like export operations, are not counted in the update sequence.) A USN counter keeps track of the most recently assigned USN.

3.5.1.1.1. Local and Global USNs

The USN is evaluated globally, for the entire database, not for the single entry. The USN is similar to the change sequence number for replication and synchronization, in that it simply ticks upward to track any changes in the database or directory. However, the entry USN is maintained separately from the CSNs, and USNs are not replicated.

The entry shows the change number for the last modification to that entry in the ***entryUSN*** operational attribute. (For details on running a search for operational attributes, see Section 10.5.7, “Searching for Operational Attributes”.)

Example 3.2. Example Entry USN

```
dn: uid=jsmith,ou=People,dc=example,dc=com
mail: jsmith@example.com
uid: jsmith
givenName: John
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: Smith
```

```
cn: John Smith
userPassword: {SSHA}EfhKCI4iKl/ipZMsWIITQatz7v2lUnptxwZ/pw==
entryusn: 1122
```

The USN Plug-in has two modes, local mode and global mode:

- In local mode, each back end database has an instance of the USN Plug-in with a USN counter specific to that back end database. This is the default setting.
- In global mode, there is a global instance of the USN Plug-in with a global USN counter that applies to changes made to the entire directory.

When the USN Plug-in is set to local mode, results are limited to the local back end database. When the USN Plug-in is set to global mode, the returned results are for the entire directory.

The root DSE shows the most recent USN assigned to any entry in the database in the ***lastusn*** attribute. When the USN Plug-in is set to local mode, so each database has its own local USN counter, the ***lastUSN*** shows both the database which assigned the USN and the USN:

```
lastusn;database_name:USN
```

For example:

```
lastusn;example1: 2130
lastusn;example2: 2070
```

In global mode, when the database uses a shared USN counter, the ***lastUSN*** attribute shows the latest USN only:

```
lastusn: 4200
```

3.5.1.1.2. Importing USN Entries

When entries are imported, the USN Plug-in uses the ***nsslapd-entryusn-import-initval*** attribute to check if the entry has an assigned USN. If the value of ***nsslapd-entryusn-import-initval*** is numerical, the imported entry will use this numerical value as the entry's USN. If the value of ***nsslapd-entryusn-import-initval*** is not numerical, the USN Plug-in will use the value of the ***lastUSN*** attribute and increment it by one as the USN for the imported entry.

3.5.1.2. Configuring the USN Plug-in

The USN Plug-in must be enabled for USNs to be recorded on entries, as described in Section 1.8.1, “Enabling Plug-ins in the Directory Server Console”. The plug-in can be enabled through the Directory Server Console or through the command line. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=USN,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

Then restart the server to apply the changes.

3.5.1.3. Enabling Global USN

The ***nsslapd-entryusn-global*** configuration parameter controls whether the USN Plug-in runs in local mode or global mode. The default is set to use a local USN counter, and the ***nsslapd-entryusn-global*** attribute value is off. To change whether the USN counter is in local or global mode, modify the ***nsslapd-entryusn-global*** attribute. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=USN,cn=plugins,cn=config
changetype: modify
replace: nsslapd-entryusn-global
nsslapd-entryusn-global: on
```

The USN Plug-in must be enabled before the ***nsslapd-entryusn-global*** configuration parameter is turned on.

3.5.1.4. Cleaning up USN Tombstone Entries

The USN Plug-in moves entries to tombstone entries when the entry is deleted. If replication is enabled, then separate tombstone entries are kept by both the USN and Replication Plug-ins. Both tombstone entries are deleted by the replication process, but for server performance, it can be beneficial to delete the USN tombstones before converting a server to a replica or to free memory for the server.

The **usn-tombstone-cleanup.pl** command deletes USN tombstone entries for a specific database back end or specific suffix. Optionally, it can delete all of tombstone entries up to a certain USN. For example:

```
/usr/lib64/dirsrv/instance_name/usn-tombstone-cleanup.pl -D "cn=directory manager" -w secret -s
"ou=people,dc=example,dc=com" -m 1100
```

Either the back end must be specified using the **-n** option or the suffix, using the **-s** option. If both are given, then the suffix in the **-s** option is used.

The options for **usn-tombstone-cleanup.pl** command are listed in Table 3.5, "usn-tombstone-cleanup.pl Options". More details for this tool are in the *Configuration and Command-Line Tool Reference* .

Table 3.5. usn-tombstone-cleanup.pl Options

Option	Description
-D *rootdn*	Gives the user DN with **root** permissions, such as Directory Manager. The default is the DN of the Directory Manager, which is read from the ***nsslapd-root*** attribute under **cn=config**.
-m *maximum_USN*	Sets the upper bound for entries to delete. All tombstone entries with an ***entryUSN*** value up to the specified maximum (inclusive) are deleted, but not past that USN value. If no maximum USN value is set, then all back end tombstone entries are deleted.
-n *backendInstance*	Gives the name of the database containing the entries to clean (delete).
-s *suffix*	Gives the name of the suffix containing the entries to clean (delete).
-w *password*	The password associated with the user DN.

3.5.2. Tracking Entry Modifications through Operational Attributes

The Directory Server can maintain some special operational attributes for every directory entry, showing basic creation and modification information:

- *creatorsName*. The distinguished name of the person who initially created the entry.
- *createTimestamp*. The timestamp for when the entry was created in GMT (Greenwich Mean Time) format.
- *modifiersName*. The distinguished name of the person who last modified the entry.
- *modifyTimestamp*. The timestamp for when the entry was last modified in GMT format.

Operational attributes are not returned in default directory searches and must be explicitly requested, as described in Section 10.5.7, "Searching for Operational Attributes".

NOTE

When a database link is used by a client application to create or modify entries, the ***creatorsName*** and ***modifiersName*** attributes do not reflect the real creator or modifier of the entries. These attributes contain the name of the administrator who is granted proxy authorization rights on the remote server.

The access logs, however, will show both the proxy user (as **dn**) and the real user (as the **authzid** entity). For example:

```
[23/May/2011:18:13:56 +051800] conn=1175 op=0 BIND dn="cn=proxy
admin,ou=people,dc=example,dc=com" method=128 version=3
[23/May/2011:18:13:56 +051800] conn=1175 op=0 RESULT err=0 tag=97 nentries=0 etime=0
dn="cn=proxy admin,ou=people,dc=example,dc=com"
[23/May/2011:18:13:56 +051800] conn=1175 op=1 SRCH base="dc=example,dc=com" scope=2
filter="(objectClass=*)" attrs=ALL authzid="uid=jsmith,ou=people,dc=example,dc=com"
```

For information on proxy authorization, see Section 2.3.1.2.2, “Providing Bind Credentials”.

To enable the Directory Server to track when entries are created or modified:

1. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
2. Select the **Settings** tab in the right pane.
3. Select the **Track Entry Modification Times** check box.

The server adds the ***creatorsName***, ***createTimestamp***, ***modifiersName***, and ***modifyTimestamp*** attributes to every newly created or modified entry.

4. Open the **Tasks** tab, and click **Restart Directory Server**.

NOTE

The Directory Server *must* be restarted for the changes to take effect.

3.5.3. Tracking the Bind DN for Plug-in Initiated Updates

One change to an entry can trigger other, automatic changes across the directory tree. When a user is deleted, for example, that user is automatically removed from any groups it belonged to by the Referential Integrity Plug-in.

The initial action is shown in the entry as being performed by whatever user account is bound to the server, but all related updates (by default) are shown as being performed by the plug-in, with no information about which user initiated that update. For example, using the MemberOf Plug-in to update user entries with group membership, the update to the group account is shown as being performed by the bound user, while the edit to the user entry is shown as being performed by the MemberOf Plug-in:

```
dn: cn=my_group,ou=groups,dc=example,dc=com
modifiersname: uid=jsmith,ou=people,dc=example,dc=com

dn: uid=bjensen,ou=people,dc=example,dc=com
modifiersname: cn=memberOf plugin,cn=plugins,cn=config
```

The ***nsslapd-plugin-binddn-tracking*** attribute allows the server to track which user originated an update operation, as well as the internal plug-in which actually performed it. The bound user is shown in the ***modifiersname*** and ***creatorsname*** operational attributes, while the plug-in which performed it is shown in the ***internalModifiersname*** and ***internalCreatorsname*** operational attributes. For example:

```
dn: uid=bjensen,ou=people,dc=example,dc=com
modifiersname: uid=jsmith,ou=people,dc=example,dc=com
internalModifiersname: cn=memberOf plugin,cn=plugins,cn=config
```

The ***nsslapd-plugin-binddn-tracking*** attribute tracks and maintains the relationship between the bound user and any updates performed for that connection.

NOTE

The ***internalModifiersname*** and ***internalCreatorsname*** attributes always show a plug-in as the identity. This plug-in could be an additional plug-in, such as the MemberOf Plug-in. If the change is made by the core Directory Server, then the plug-in is the database plug-in, **cn=ldbm database,cn=plugins,cn=config**.

The ***nsslapd-plugin-binddn-tracking*** attribute is disabled by default. To allow the server to track operations based on bind DN, enable that attribute using **ldapmodify**:

```
[jsmith@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-plugin-binddn-tracking
nsslapd-plugin-binddn-tracking: on
```

3.5.4. Tracking Password Change Times

Password change operations are normally treated as any other modification to an entry, so the update time is recorded in the ***lastModified*** operational attrobite. However, there can be times when the time of the last password change needs to be recorded separately, to make it easier to update passwords in Active Directory synchronization or to connect with other LDAP clients.

The ***passwordTrackUpdateTime*** attribute within the password policy tells the server to record a timestamp for the last time that the password was updated for an entry. The password change time itself is stored as an operational attribute on the user entry, ***pwdUpdateTime*** (which is separate from the ***modifyTimestamp*** or ***lastModified*** operational attributes).

The ***passwordTrackUpdateTime*** attribute can be set as part of the global password policy or on a subtree or user-level policy, depending on what clients need to access the password change time. Setting password policies is described in Section 14.1, “Managing the Password Policy” .

3.6. MAINTAINING REFERENTIAL INTEGRITY

Referential Integrity is a database mechanism that ensures relationships between related entries are maintained. In the Directory Server, the Referential Integrity can be used to ensure that an update to one entry in the directory is correctly reflected in any other entries that reference to the updated entry.

For example, if a user's entry is removed from the directory and referential integrity is enabled, the server also removes the user from any groups of which the user is a member. If referential integrity is not enabled, the user remains a member of the group until manually removed by the administrator. This is an important feature if you are integrating the Directory Server with other products that rely on the directory for user and group management.

3.6.1. How Referential Integrity Works

When the Referential Integrity Plug-in is enabled, it performs integrity updates on specified attributes immediately after a delete or rename operation. By default, the Referential Integrity Plug-in is disabled.

NOTE

The Referential Integrity Plug-in should only be enabled on one supplier replica in a multi-master replication environment to avoid conflict resolution loops. When enabling the plug-in on servers issuing chaining requests, be sure to analyze performance resource and time needs, as well as your integrity needs. Integrity checks can be time-consuming and draining on memory and CPU.

When a user or group entry is deleted, updated, renamed, or moved within the directory, the operation is logged to the Referential Integrity log file. For the distinguished names (DN) in the log file, Directory Server searches and updates in intervals the attributes set in the plug-in configuration:

- For entries, marked in the log file as deleted, the corresponding attribute in the directory is deleted.
- For entries, marked in the log file as updated, the corresponding attribute in the directory is updated.
- For entries, marked in the log file as renamed or moved, the value of the corresponding attribute in the directory is renamed.

By default, when the Referential Integrity Plug-in is enabled, it performs integrity updates on the ***member***, ***uniquemember***, ***owner***, and ***seeAlso*** attributes immediately after a delete or rename operation. However, the behavior of the Referential Integrity Plug-in can be configured to suit the needs of the directory in several different ways:

- Record referential integrity updates in the replication changelog.
- Modify the update interval.
- Select the attributes to which to apply referential integrity.
- Disable referential integrity.

All attributes used in referential integrity *must* be indexed for presence, equality, and subtring; not indexing those attributes results poor server performance for modify and delete operations.

```
nsIndexType: pres
nsIndexType: eq
nsIndexType: sub
```

See Section 9.2, “Creating Standard Indexes” for more information about checking and creating indexes.

3.6.2. Using Referential Integrity with Replication

There are certain limitations when using the Referential Integrity Plug-in in a replication environment:

- *Never* enable it on a dedicated consumer server (a server that contains only read-only replicas).
- *Never* enable it on a server that contains a combination of read-write and read-only replicas.
- It is possible to enable it on a supplier server that contains only read-write replicas.
- With multi-master replication, enable the plug-in on just one supplier.

If the replication environment satisfies the all of those condition, you can enable the Referential Integrity Plug-in.

1. Enable the Referential Integrity Plug-in as described in Section 3.6.3, “Enabling and Disabling Referential Integrity”.

2. Configure the plug-in to record any integrity updates in the changelog.
3. Ensure that the Referential Integrity Plug-in is disabled on all consumer servers.

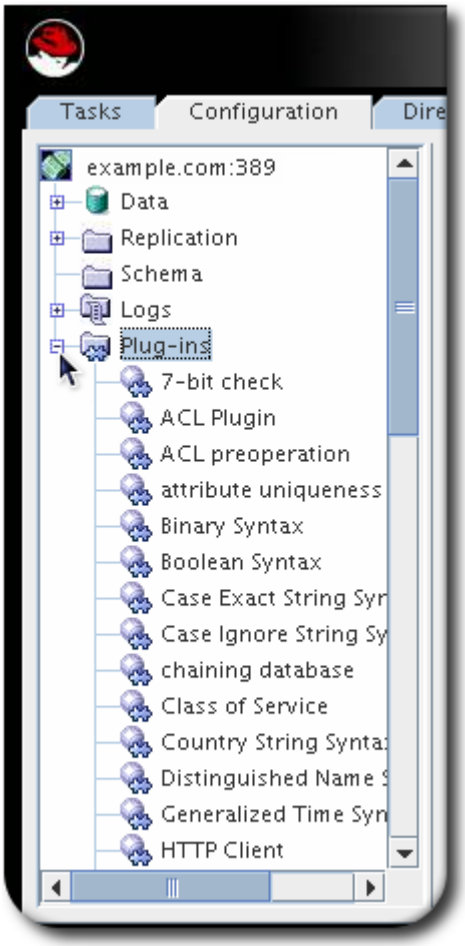
NOTE

Because the supplier server sends any changes made by the Referential Integrity Plug-in to consumer servers, it is unnecessary to run the Referential Integrity Plug-in on consumer servers.

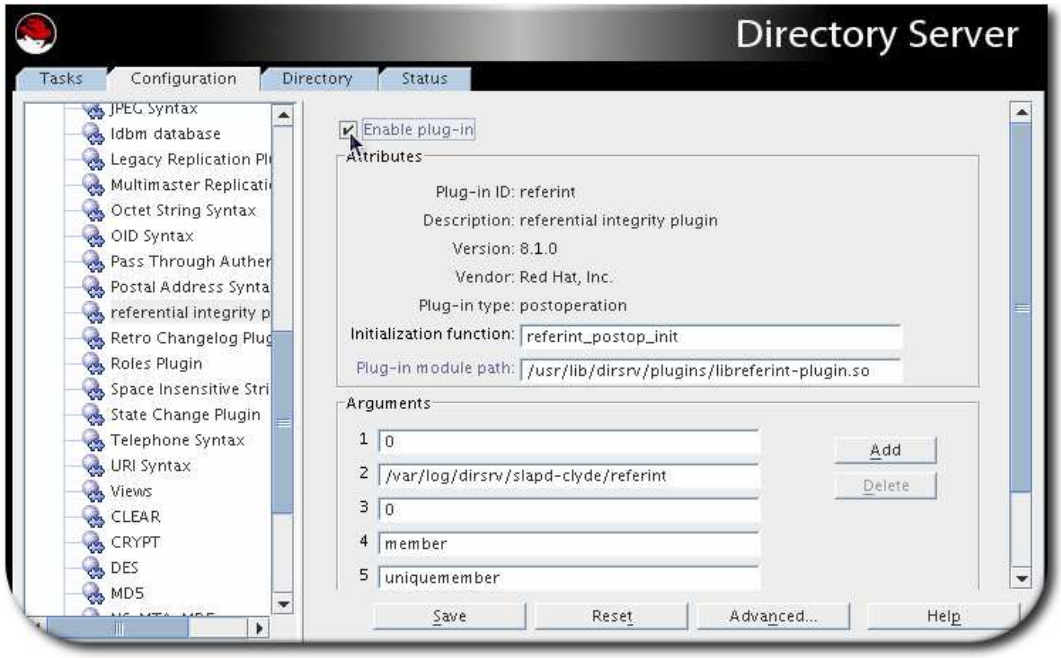
3.6.3. Enabling and Disabling Referential Integrity

3.6.3.1. Enabling and Disabling Referential Integrity in the Console

1. Select the **Configuration** tab, and expand the **Plugins** folder.
2. Select **Referential Integrity Postoperation Plug-in** from the list.

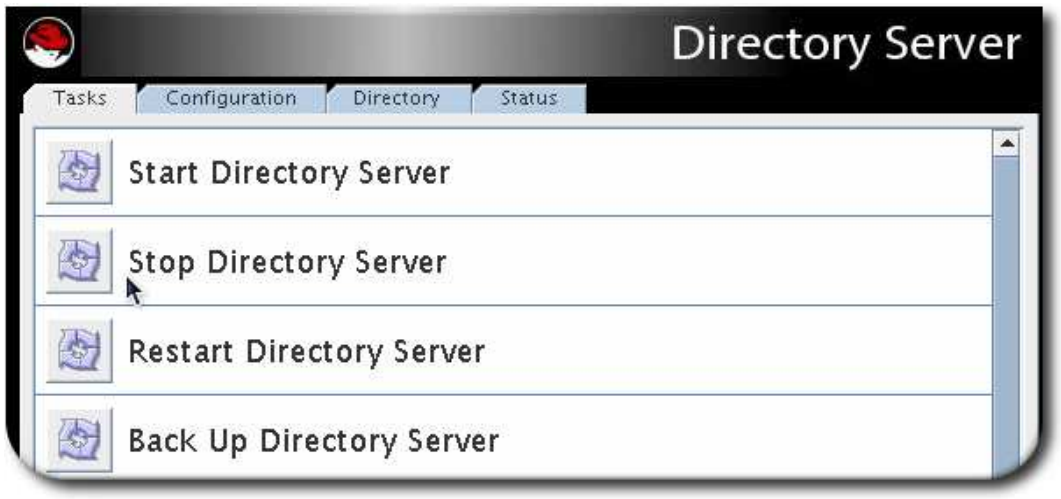


3. Check the **Enable plugin** check box to enable the plug-in; clear it to disable it.



4. Fill in the correct path to the plug-in by default; plug-ins are located in **/usr/lib64/dirsrv/plugins**.

5. Restart the Directory Server to apply the changes. In the **Tasks** tab, select **Restart the Directory Server**.



3.6.3.2. Enabling and Disabling Referential Integrity from the Command Line

To disable or enable the Referential Integrity Plug-in:

1. Use **ldapmodify** to edit the value of the ***nsslapd-pluginEnabled*** attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=referential integrity postoperation,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Then restart the server.

```
service dirsrv restart
```

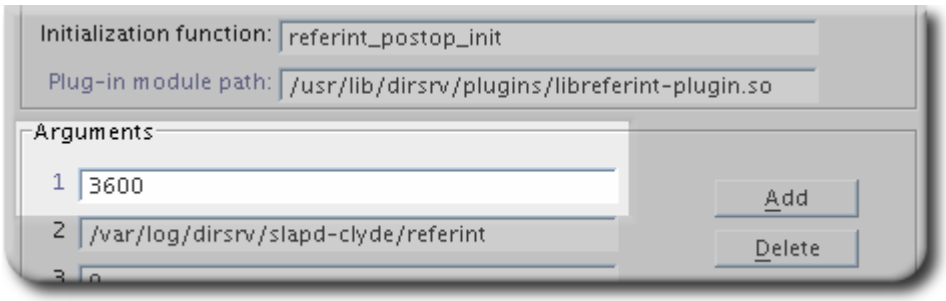
3.6.4. Modifying the Update Interval

By default, the server makes referential integrity updates immediately after a delete or a **modrdn** operation. To reduce the impact this operation has on your system, increase the amount of time between updates. Although there is no maximum update interval, the following intervals are commonly used:

- Update immediately
- 90 seconds
- 3600 seconds (updates occur every hour)
- 10,800 seconds (updates occur every 3 hours)
- 28,800 seconds (updates occur every 8 hours)
- 86,400 seconds (updates occur once a day)
- 604,800 seconds (updates occur once a week)

3.6.4.1. Modifying the Update Interval from the Console

1. Select the **Configuration** tab, and expand the **Plugins** folder. Select the **Referential Integrity Postoperation Plug-in**.
2. In the arguments list, replace the value in the first text box with the appropriate time interval.



3. Restart the Directory Server to apply the changes. In the **Tasks** tab, select **Restart the Directory Server**.

3.6.4.2. Modifying the Update Interval from the Command Line

1. Use **ldapmodify** to edit the value of the ***nsslapd-pluginarg*** attribute. For example:

   The first argument listed sets the update interval for referential integrity checks. To change the interval, replace the ***nsslapd-pluginarg0*** attribute.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=referential integrity postoperation,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: 600
```

2. Then restart the server.

```
service dirsrv restart
```

3.6.5. Modifying the Attribute List

3.6.5.1. Modifying the Attribute List from the Console

By default, the Referential Integrity Plug-in is set up to check for and update the ***member***, ***uniquemember***, ***owner***, and ***seeAlso*** attributes. You can add or delete attributes to be updated through the Directory Server Console, such as adding the ***nsroledn*** attribute if roles are being used.

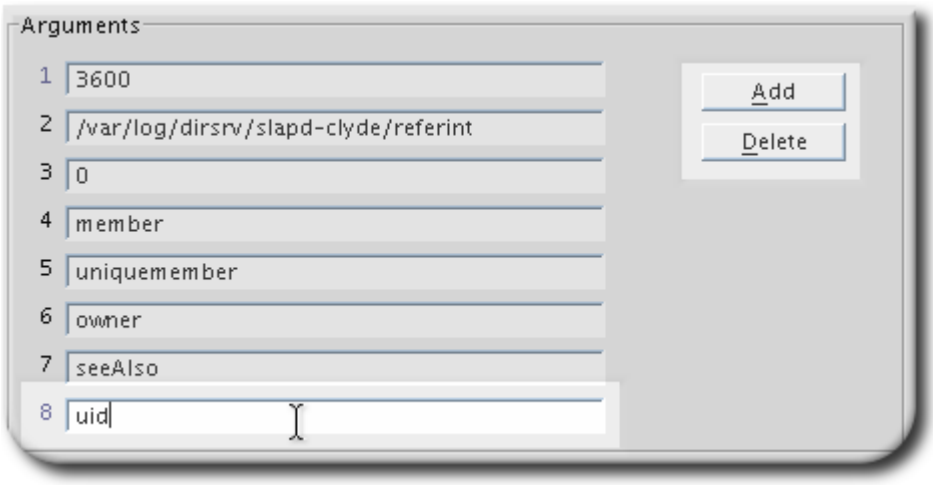
NOTE

Keep in mind that any attribute specified in the Referential Integrity Plug-in parameter list *must* have equality indexing on all databases. Otherwise, the plug-in scans every entry of the databases for matching the deleted or modified DN, degrading performance severely. If you add an attribute, ensure that it is indexed in all the back ends.

NOTE

Improve the performance by removing any unused attributes from the list.

1. Select the **Configuration** tab, and expand the **Plugins** folder. Select the **Referential Integrity Postoperation Plug-in**.
2. In the **Arguments** section, use the **Add** and **Delete** buttons to modify the attributes in the list.



3. Restart the Directory Server to apply the changes. In the **Tasks** tab, select **Restart the Directory Server**.

NOTE

All attributes used in referential integrity *must* be indexed for presence and equality; not indexing those attributes results poor server performance for modify and delete operations. See Section 9.2, "Creating Standard Indexes" for more information about checking and creating indexes.

3.6.5.2. Modifying the Attribute List from the Command Line

By default, the Referential Integrity plug-in is set up to check for and update the ***member***, ***uniquemember***, ***owner***, and ***seeAlso*** attributes.

To enable shared configuration entries, set the **nsslapd-pluginConfigArea** attribute:

```
nsslapd-pluginConfigArea:entry_DN
```

All the configuration attribute settings, for example adding or removing a shared entry, are dynamic and do not require a server restart to take effect.

The following example uses the **pluginarg*** attributes:

```
nsslapd-pluginarg0: 0
nsslapd-pluginarg1: /var/log/dirsrv/slapd-localhost/referint
nsslapd-pluginarg2: 0
nsslapd-pluginarg3: member
nsslapd-pluginarg4: uniquemember
nsslapd-pluginarg5: owner
nsslapd-pluginarg6: seeAlso
```

Referential Integrity plug-in parameter descriptions:

Legacy-style parameter	Description

Legacy-style parameter	Description
nsslapd-pluginarg0	Sets the update delay: • **-1**: No check for referential integrity is performed. • **0**: The check for referential integrity is performed immediately. • Positive integer value: Represents the interval in seconds for performing the referential integrity check.
nsslapd-pluginarg1	Sets the path to the log file.
nsslapd-pluginarg2	
nsslapd-pluginarg[3-10]	Sets the attributes on which the plug-in performs integrity updates.

NOTE

Keep in mind that any attribute specified in the Referential Integrity Plug-in parameter list *must* have equality indexing on all databases. Otherwise, the plug-in scans every entry of the databases for matching the deleted or modified DN, degrading performance severely. If you add an attribute, ensure that it is indexed in all the back ends.

CHAPTER 4. POPULATING DIRECTORY DATABASES

Databases contain the directory data managed by the Red Hat Directory Server.

4.1. IMPORTING DATA

Directory Server can populate a database with data in one of two ways: by importing data (either through the Directory Server Console or using the import tools) or by initializing a database for replication.

Table 4.1, "Import Method Comparison" describes the differences between an import and initializing databases.

Table 4.1. Import Method Comparison

Action	Import	Initialize Database
Overwrites database	No	Yes
LDAP operations	Add, modify, delete	Add only
Performance	More time-consuming	Fast
Partition specialty	Works on all partitions	Local partitions only
Response to server failure	Best effort (all changes made up to the point of the failure remain)	Atomic (all changes are lost after a failure)
LDIF file location	Local to Console	Local to Console or local to server
Imports configuration information (**cn=config**)	Yes	No

4.1.1. Importing Entries with Large Attributes

The ***nsslapd-cachememsize*** attribute defines the size allowed for the entry cache.

The import buffer is automatically set to 80% of the cache memory size setting. If the memory cache is 1GB, for example, then the import buffer is 800MB.

When importing a very large database or entries with large attributes (often with values like binary data like certificate chains, CRLs, or images), then set the ***nsslapd-cachememsize*** attribute high enough so that the import buffer has enough memory to process the entries.

4.1.2. Importing Large Numbers of Entries

When there are a large number of entries to be imported, the operating system itself may hit performance limits on what it allows the Directory Server to do. This is particularly true on x86 systems. This can cause import operations to fail because of resource constraints.

If necessary, set the system **ulimit** value to the maximum number of allows processes for the system user.

For example:

```
[root@server ~]# ulimit -u 4096
```

Then run the import operation.

4.1.3. Setting EntryUSN Initial Values During Import

Entry update sequence numbers (USNs) are not preserved when entries are exported from one server and imported into another. As Section 3.5.1, "Tracking Modifications to the Database through Update Sequence Numbers" explains, entry USNs are assigned for operations that happen on a local server, so it does not make sense to import those USNs onto another server.

However, it is possible to configure an initial entry USN value for entries when importing a database or initializing a database (such as when a replica is initialized for replication). This is done by setting the ***nsslapd-entryusn-import-initval*** attribute, which sets a starting USN for all imported entries.

There are two possible values for ***nsslapd-entryusn-import-initval***:

- An integer, which is the explicit start number used for every imported entry.
- *next*, which means that every imported entry uses whatever the highest entry USN value was on the server before the import operation, incremented by one.

If ***nsslapd-entryusn-import-initval*** is not set, then all entry USNs begin at zero.

For example, if the highest value on the server is 1000 before the import or initialization operation, and the ***nsslapd-entryusn-import-initval*** value is next, then every imported entry is assigned a USN of 1001:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x "(cn=*)" entryusn

dn: dc=example,dc=com
entryusn: 1001
dn: ou=Accounting,dc=example,dc=com
entryusn: 1001
dn: ou=Product Development,dc=example,dc=com
entryusn: 1001
...
dn: uid=jsmith,ou=people,dc=example,dc=com
entryusn: 1001
...
```

To set an initial value for entry USNs, simply add the ***nsslapd-entryusn-import-initval*** attribute to the server into which data are being imported or to the master server which will perform the initialization.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 389 -h
server.example.com -x

dn: cn=config
changetype: modify
add: nsslapd-entryusn-import-initval
nsslapd-entryusn-import-initval: next
```

NOTE

In multi-master replication, the ***nsslapd-entryusn-import-initval*** attribute is *not* replicated between servers. This means that the value must be set specifically on whichever supplier server is being used to initialize a replica.

For example, if Supplier1 has ***nsslapd-entryusn-import-initval*** set to *next* and is used to initialize a replica, then the entry USNs for imported entries have the highest value plus one. If Supplier2 does not have ***nsslapd-entryusn-import-initval*** set and is used to initialize a replica, then all entry USNs for imported entries begin at zero — even if Supplier1 and Supplier 2 have a multi-master replication agreement between them.

4.1.4. Importing a Database from the Console

When performing an import operation from the Directory Server Console, an **ldapmodify** operation is executed to append data, as well as to modify and delete entries. The operation is performed on all of the databases managed by the Directory Server and on remote databases to which the Directory Server has a configured database link.

Import operations can be run on a server instance that is local to the Directory Server Console or on a different host machine (a remote import operation).

You must be logged in as the Directory Manager in order to perform an import.

NOTE

The LDIF files used for import operations must use UTF-8 character set encoding. Import operations do not convert data from local character set encoding to UTF-8 characterset encoding.

WARNING

All imported LDIF files must also contain the root suffix.

To import data from the Directory Server Console:

1. Select the **Tasks** tab. Scroll to the bottom of the screen, and select **Import Database**.

   Alternatively, open the **Configuration** tab and select **Import** from the **Console** menu.

2. In the **Import Database** dialog box, enter the full path to the LDIF file to import in the **LDIF file** field, or click **Browse** to select the file to import.

   If the Console is running on a machine remote to the directory, the field name appears as **LDIF file (on the machine running the Console)**. When browsing for a file, you are not browsing the current directory for the Directory Server host, but the filesystem of the machine running the Console.

   When importing a database through a remote Console, *do not* use a relative path to the database. For remote imports, the operation fails with the error *Cannot write to file...* if a relative path is given for the file. Always use an absolute path for remote import operations.

3. In the **Options** box, select one or both of the following options:

   - *Add Only.* The LDIF file may contain modify and delete instructions in addition to the default add instructions. For the server to ignore operations other than add, select the **Add only** check box.

   - *Continue on Error.* Select the **Continue on error** check box for the server to continue with the import even if errors occur. For example, use this option to import an LDIF file that contains some entries that already exist in the database in addition to new ones. The server notes existing entries in the rejects file while adding all new entries.

4. In the **File for Rejects** field, enter the full path to the file in which the server is to record all entries it cannot import, or click **Browse** to select the file which will contain the rejects.

   A reject is an entry which cannot be imported into the database; for example, the server cannot import an entry that already exists in the database or an entry that has no parent object. The Console will write the error message sent by the server to the rejects file.

   Leaving this field blank means the server will not record rejected entries.

The server performs the import and also creates indexes.

NOTE

Trailing spaces are dropped during a remote Console import but are preserved during both local Console or **ldif2db** import operations.

4.1.5. Initializing a Database from the Console

The existing data in a database can be overwritten by initializing databases.

You must be logged in as the **Directory Manager** in order to initialize a database because an LDIF file that contains a root entry cannot be imported into a database except as the Directory Manager (root DN). Only the Directory Manager has access to the root entry, such as **dc=example,dc=com**.

WARNING

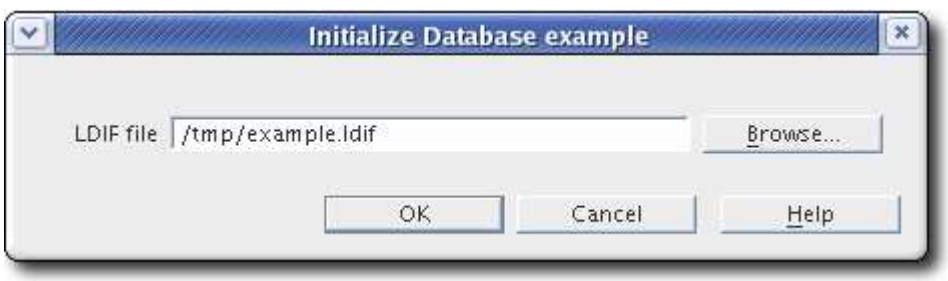
When initializing databases from an LDIF file, be careful not to overwrite the **o=NetscapeRoot** suffix unless you are restoring data. Otherwise, initializing the database deletes information and may require re-installing the Directory Server.

To initialize a database using the Directory Server Console:

1. Select the **Configuration** tab.
2. Expand the **Data** tree in the left navigation pane. Expand the suffix of the database to initialize, then click the database itself.
3. Right-click the database, and select **Initialize Database**.

Alternatively, select **Initialize Database** from the **Object** menu.

4. In the **LDIF file** field, enter the full path to the LDIF file to import, or click **Browse**.



5. If the Console is running from a machine local to the file being imported, click **OK** and proceed with the import immediately. If the Console is running from a machine remote to the server containing the LDIF file, select one of the following options, then click **OK**:

   - *From local machine.* Indicates that the LDIF file is located on the local machine.
   - *From server machine.* Indicates that the LDIF file is located on a remote server.

   The default LDIF directory is **/var/lib/dirsrv/slapd-*instance_name*/ldif**.

4.1.6. Importing from the Command Line

There are four methods for importing data through the command line:

- *Using ldif2db.* This import method overwrites the contents of the database and requires the server to be stopped; see Section 4.1.6.1, "Importing Using the ldif2db Command-Line Script" .
- *Using ldif2db.pl.* This import method overwrites the contents of the database while the server is still running; see Section 4.1.6.2, "Importing Using the ldif2db.pl Perl Script" .
- *Using ldif2ldap.* This method appends the LDIF file through LDAP. This method is useful to append data to all of the databases; see Section 4.1.6.3, "Importing Using the ldif2ldap Command-Line Script".
- *Creating a cn=tasks entry.* This method creates a temporary task entry which automatically launches an import operation. This is functionally like running **ldif2db**. See Section 4.1.6.4, "Importing through the cn=tasks Entry".

NOTE

The LDIF files used for import operations must use UTF-8 character set encoding. Import operations do not convert data from local character set encoding to UTF-8 characterset encoding.

WARNING

All imported LDIF files must also contain the root suffix.

NOTE

To import a database that has been encrypted, use the **-E** option with the script. See Section 2.2.3.6, "Exporting and Importing an Encrypted Database" for more information.

4.1.6.1. Importing Using the ldif2db Command-Line Script

The **ldif2db** script overwrites the data in the specified database. Also, the script requires that the Directory Server be stopped when the import begins.

By default, the script first saves and then merges any existing **o=NetscapeRoot** configuration information with the **o=NetscapeRoot** configuration information in the files being imported.

WARNING

This script overwrites the data in the database.

To import an LDIF:

1. Stop the server.

```
[root@server ~]# service dirsrv stop instance
```

2. Run the **ldif2db** command-line script.

```
[root@server ~]# /usr/lib64/dirsrv/slapd-instance_name/ldif2db -n Database1 -i
/var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif -i /var/lib/dirsrv/slapd-instance_name/ldif/demo2.ldif
```

On 32-bit installations, the **ldif2db** script is located in the **/usr/lib64/dirsrv/slapd-*instance_name*/** directory.

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.

WARNING

If the database specified in the **-n** option does not correspond with the suffix contained by the LDIF file, all of the data contained by the database is deleted, and the import fails. Make sure that the database name is not misspelled.

3. Start the server.

```
[root@server ~]# service dirsrv start instance
```

Table 4.2. ldif2db Parameters

Option	Description
-i	Specifies the full path name of the LDIF files to be imported. This option is required. To import more than one LDIF file at a time, use multiple **-i** arguments. When multiple files are imported, the server imports the LDIF files in the order which they are specified from the command line.
-n	Specifies the name of the database to which to import the data.

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.

4.1.6.2. Importing Using the ldif2db.pl Perl Script

As with the **ldif2db** script, the **ldif2db.pl** script overwrites the data in the specified database. This script requires the server to be running in order to perform the import.

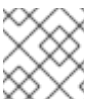
WARNING

This script overwrites the data in the database.

Run the **ldif2db.pl** script.

```
[root@server ~]# ldif2db.pl -D "cn=Directory Manager" -w secret -i
/var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif -n Database1
```

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.



NOTE

You do not need **root** privileges to run the script, but you must authenticate as the Directory Manager.

Table 4.3. ldif2db.pl Options

Option	Description
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
-i	Specifies the LDIF files to be imported. This option is required. To important multiple LDIF files at a time, use multiple **-i** arguments. When multiple files are imported, the server imports the LDIF files in the order they are specified in the command line.
-n	Specifies the name of the database to which to import the data.

4.1.6.3. Importing Using the ldif2ldap Command-Line Script

The **ldif2ldap** script appends the LDIF file through LDAP. Using this script, data are imported to all directory databases at the same time. The server must be running in order to import using **ldif2ldap**.

To import LDIF using **ldif2ldap**:

```
[root@server ~]# ldif2ldap "cn=Directory Manager" secretpwd /var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif
```

The **ldif2ldap** script requires the DN of the administrative user, the password of the administrative user, and the absolute path and filename of the LDIF files to be imported.

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.

4.1.6.4. Importing through the cn=tasks Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=import,cn=tasks,cn=config** to initiate an import operation.

As with the **ldif2db** and **ldif2db.pl** scripts, an import operation in **cn=tasks** overwrites all of the information in the database.

This task entry requires three attributes:

- A unique name (***cn***)
- The filename of the LDIF file to import (***nsFilename***)
- The name of the database into which to import the file (***nsInstance***)

It is also possible to supply the DNs of suffixes to include or exclude from the import, analogous to the **-s** and **-x** options, respectively, for the **ldif2db** and **ldif2db.pl** scripts.

The entry is simply added using **ldapmodify**, as described in Section 3.2.4, "Adding and Modifying Entries Using ldapmodify". For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example import,cn=import,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example import
nsFilename: /home/files/example.ldif
nsInstance: userRoot
nsIncludeSuffix: ou=People,dc=example,dc=com
nsExcludeSuffix: ou=Groups,dc=example,dc=com
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on the available attributes for running Directory Server import tasks under the **cn=tasks** entries.

4.2. EXPORTING DATA

LDAP Data Interchange Format (LDIF) files are used to export database entries from the Directory Server databases. LDIF is a standard format described in RFC 2849, *The LDAP Data Interchange Format (LDIF) - Technical Specification*.

Exporting data can be useful for the following:

- Backing up the data in the database.
- Copying data to another Directory Server.
- Exporting data to another application.
- Repopulating databases after a change to the directory topology.

For example, if a directory contains one database, and its contents are split into two databases, then the two new databases receive their data by exporting the contents of the old databases and importing it into the two new databases, as illustrated in Figure 4.1, "Splitting a Database Contents into Two Databases" .

NOTE

The export operations do not export the configuration information (**cn=config**), schema information (**cn=schema**), or monitoring information (**cn=monitor**).

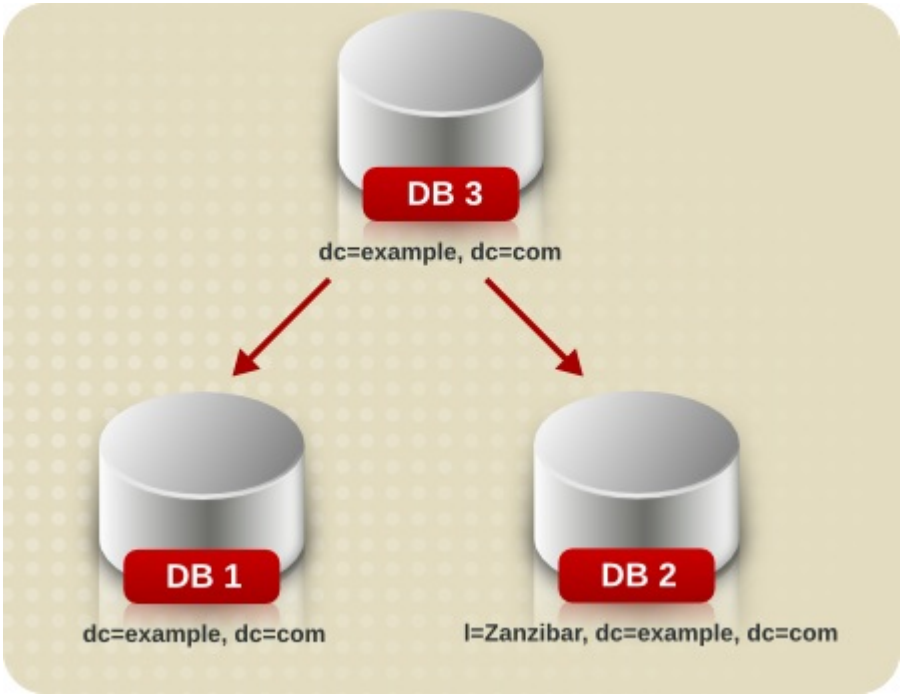


Figure 4.1. Splitting a Database Contents into Two Databases

The Directory Server Console or command-line utilities can be used to export data.

- Section 4.2.1, "Exporting Directory Data to LDIF Using the Console"
- Section 4.2.2, "Exporting a Single Database to LDIF Using the Console"
- Section 4.2.3, "Exporting to LDIF from the Command Line"

WARNING

Do not stop the server during an export operation.

4.2.1. Exporting Directory Data to LDIF Using the Console

Some or all of directory data can be exported to LDIF, depending upon the location of the final exported file. When the LDIF file is on the server, only the data contained by the databases local to the server can be exported. If the LDIF file is remote to the server, all of the databases and database links can be exported.

Export operations can be run to get data from a server instance that is local to the Directory Server Console or from a different host machine (a remote export operation).

Export directory data to LDIF from the Directory Server Console while the server is running:

1. Select the **Tasks** tab. Scroll to the bottom of the screen, and click **Export Database(s)**.

   Alternatively, select the **Configuration** tab and click the **Export from the Console** menu.

2. Enter the full path and filename of the LDIF file in the **LDIF File** field, or click **Browse** to locate the file.

   Browse is not enabled if the Console is running on a remote server. When the **Browse** button is not enabled, the file is stored in the default directory, **/var/lib/dirsrv/slapd-*instance_name*/ldif**.

3. If the Console is running on a machine remote to the server, two radio buttons are displayed beneath the **LDIF File** field.

   - Select **To local machine** to export the data to an LDIF file on the machine from which the Console is running.
   - Select **To server machine** to export to an LDIF file located on the server's machine.

4. To export the whole directory, select the **Entire database** radio button.

   To export only a single subtree of the suffix contained by the database, select the **Subtree** radio button, and then enter the name of the suffix in the **Subtree** text box. This option exports a subtree that is contained by more than one database.

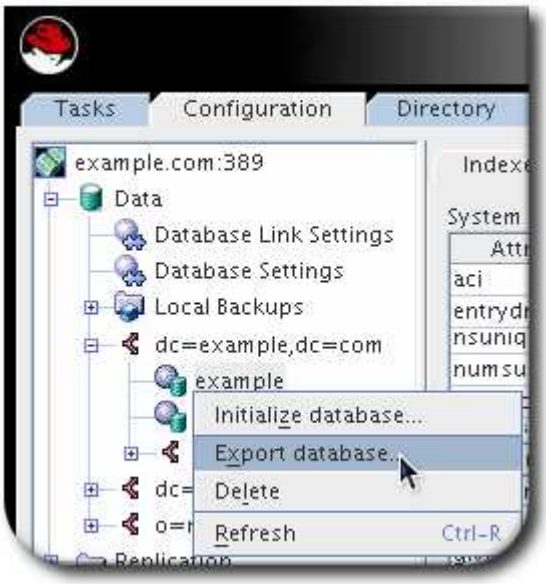
   Alternatively, click **Browse** to select a suffix or subtree.

4.2.2. Exporting a Single Database to LDIF Using the Console

It is also possible to export a single database to LDIF. Do the following while the server is running:

1. Select the **Configuration** tab.
2. Expand the **Data** tree in the left navigation pane. Expand the suffix, and select the database under the suffix.

3. Right-click the database, and select **Export Database**.



Alternatively, select **Export Database** from the **Object** menu.

4. In the **LDIF file** field, enter the full path to the LDIF file, or click **Browse**.

When the **Browse** button is not enabled, the file is stored in the default directory, **/var/lib/dirsrv/slapd-*instance_name*/ldif**.

4.2.3. Exporting to LDIF from the Command Line

There are three methods for exporting data through the command line:

- *Using db2ldif.* This method runs the command-line utility; unlike the import script, **ldif2db**, this utility can be run while the server is running.
- *Using db2ldif.pl.* This Perl script behaves the same as the **db2ldif** command-line utility and takes the same options.
- *Creating a cn=tasks entry.* This method creates a temporary task entry which automatically launches an export operation. This is functionally like running **db2ldif**, with one exception: when running **db2ldif** or **db2ldif.pl** for a replica (with a **-r** option, the server must be stopped first. The **cn=tasks** entry can be added and export replica information while the server is still running. See Section 4.2.3.2, "Exporting through the cn=tasks Entry".

4.2.3.1. Exporting a Database Using db2ldif or db2ldif.pl

Databases can be exported to LDIF using the **db2ldif** command-line script or the **db2ldif.pl** Perl script. Both of these tools export all of the database contents or a part of their contents to LDIF when the server is running or stopped.

These script take the same options.

NOTE

The **-E** option is required to export a database that has been encrypted. See Section 2.2.3.6, "Exporting and Importing an Encrypted Database" for more information.

NOTE

If the database being exported is a replica, then the server must be stopped before the export script is run and the export script must have the **-r**.

To export to LDIF from the command linerun either the **db2ldif** command-line script or the **db2ldif.pl** Perl script. For example:

```
[root@server ~]# db2ldif -n database1 -a /export/output.ldif
```

This exports the database contents to **/export/output.ldif**. If the **-a** option is not specified, then the database information is exported to **/var/lib/dirsrv/slapd-*instance_name*/ldif/***instance_name***-database1-***date***.ldif**. For example:

```
[root@server ~]# db2ldif -n database1
```

It is also possible to specify which suffixes to export, using the **-s** option. For example:

```
[root@server ~]# db2ldif -s "dc=example,dc=com"
```

The LDIF file in this case would be **/var/lib/dirsrv/slapd-*instance_name*/ldif/***instance_name***-example-2017_04_30_112718.ldif**, using the name of the suffix rather than the database.

If the suffix specified is a root suffix, such as **dc=example,dc=com**, then it is not necessary to specify the database or to use the **-n** option.

For more information about using these scripts, see the *Directory Server Configuration and Command-Line Tool Reference*.

Table 4.4. db2ldif Options

Option	Description
-n	Specifies the name of the database from which the file is being exported.
-s	Specifies the suffix or suffixes to include in the export. If the suffix is a root suffix, such as **dc=example,dc=com**, then the **-n** option is not required. There can be multiple**-s** arguments.
-a	Defines the output file to which Directory Server exports the LDIF. This file must be an absolute path. If the **-a** option is not given, the output ldif is stored in the **/var/lib/dirsrv/slapd-*instance_name*/ldif** directory and is automatically named *serverID*-**database-***YYYY_MM_DD_hhmmxx***.ldif** with the**-n** option or *serverID-firstsuffixvalue-YYYY_MM_DD_hhmmxx***.ldif** with the **-s** option.
-r	Specifies that the exported database is a consumer replica. In this case, the appropriate settings and entries are included with the LDIF to initialize the replica when the LDIF is imported.
-E	Decrypts an encrypted database so it can be exported.

4.2.3.2. Exporting through the cn=tasks Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=export,cn=tasks,cn=config** to initiate an export operation.

The export task entry requires three attributes:

- A unique name (***cn***)

- The filename of the LDIF file to which to export the database (***nsFilename***)
- The name of the database to export (***nsInstance***)

It is also possible to supply the DNs of suffixes to include or exclude from the export operation, analogous to the **-s** and **-x** options, respectively, for the **db2ldif** and **db2ldif.pl** scripts. Additionally, if the database is a replica, then the appropriate replica information can be included to initialize the new consumer when the LDIF is imported; this is set in the ***nsExportReplica***, corresponding to the **-r** option.

The entry is simply added using **ldapmodify**, as described in Section 3.2.4, "Adding and Modifying Entries Using ldapmodify". For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example export,cn=export,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example export
nsInstance: userRoot
nsFilename: /home/files/example.ldif
nsExportReplica: true
nsIncludeSuffix: ou=People,dc=example,dc=com
nsExcludeSuffix: ou=Groups,dc=example,dc=com
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on the available attributes for running Directory Server export tasks under the **cn=tasks** entries.

4.3. BACKING UP AND RESTORING DATA

Databases can be backed up and restored using the Directory Server Console or a command-line script.

- Section 4.3.1, "Backing up All Databases"
- Section 4.3.2, "Backing up the dse.ldif Configuration File"
- Section 4.3.3, "Restoring All Databases"
- Section 4.3.4, "Restoring a Single Database"
- Section 4.3.5, "Restoring Databases That Include Replicated Entries"
- Section 4.3.6, "Restoring the dse.ldif Configuration File"

WARNING

Do not stop the server during a backup or restore operation.

NOTE

The changelog database is backed up with the regular server database.

4.3.1. Backing up All Databases

The following procedures describe backing up all of the databases in the directory using the Directory Server Console and from the command line.

NOTE

These backup methods cannot be used to back up the data contained by databases on a remote server that are chained using database links.

4.3.1.1. Backing up All Databases from the Console

When backing up databases from the Directory Server Console, the server copies all of the database contents and associated index files to a backup location. A backup can be performed while the server is running.

To back up databases from the Directory Server Console:

1. Select the **Tasks** tab.
2. Click **Back Up Directory Server**.

3. Enter the full path of the directory to store the backup file in the **Directory** text box, or click **Use default**, and the server provides a name for the backup directory.

If the Console is running on the same machine as the directory, click **Browse** to select a local directory.

With the default location, the backup files are placed in **/var/lib/dirsrv/slapd-*instance_name*/bak**. By default, the backup directory name contains the name of the server instance and the time and date the backup was created (*instance_name*-*YYYY_MM_DD_hhmmss*).

4.3.1.2. Backing up All Databases from the Command Line

Databases can be backed up from the command line using either the **db2bak** command-line script or the **db2bak** Perl script. The command-line script works when the server is running or when the server is stopped; the Perl script can only be run when the server is running.

IMPORTANT

If the database being backed up is a master database, meaning it keeps a changelog, then it must be backed up using the **db2bak.pl** Perl script or using the Directory Server Console if the server is kept running. The changelog only writes its RUV entries to the database when the server is shut down; while the server is running, the changelog keeps its changes in memory. For the Perl script and the Console, these changelog RUVs are written to the database before the backup process runs. However, that step is not performed by the command-line script.

The **db2bak** should not be run on a running master server. Either use the Perl script or stop the server before performing the backup.

Configuration information *cannot* be backed up using this backup method. For information on backing up the configuration information, see Section 4.3.2, "Backing up the dse.ldif Configuration File".

To back up the directory from the command line using the **db2bak.pl** script, run the **db2bak.pl** Perl script, specifying the backup filename and directory.

```
[root@server ~]# /usr/lib[64]/dirsrv/slapd-example/db2bak.pl
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2017_04_30_16_27_56
```

The backup directory where the server saves the backed up databases can be specified with the script. If a directory is not specified, the backup file is stored in **/var/lib/dirsrv/slapd-*instance_name*/bak**. By default, the backup directory is named with the Directory Server instance name and the date of the backup (*serverID*-*YYYY_MM_DD_hhmmss*).

For more information about using these scripts, see the *Directory Server Configuration and Command-Line Tool Reference*.

4.3.1.3. Backing up the Database through the cn=tasks Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=backup,cn=tasks,cn=config** to initiate a backup operation.

The backup task entry requires three attributes:

- A unique name (***cn***).
- The directory to write the backup file to (***nsArchiveDir***). The backup file is named with the Directory Server instance name and the date of the backup (*serverID-YYYY_MM_DD_hhmmss*).
- The type of database (***nsDatabaseType***); the only option is **ldbm database**.

The entry is simply added using **ldapmodify**, as described in Section 3.2.4, "Adding and Modifying Entries Using ldapmodify". For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example backup,cn=backup,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example backup
nsArchiveDir: /export/backups/
nsDatabaseType: ldbm database
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on the available attributes for running tasks to back up databases under the **cn=tasks** entries.

4.3.2. Backing up the dse.ldif Configuration File

Directory Server automatically backs up the **dse.ldif** configuration file. When the Directory Server is started, the directory creates a backup of the **dse.ldif** file automatically in a file named **dse.ldif.startOK** in the **/etc/dirsrv/slapd-*instance_name*** directory.

When the **dse.ldif** file is modified, the file is first backed up to a file called **dse.ldif.bak** in the **/etc/dirsrv/slapd-*instance_name*** directory before the directory writes the modifications to the **dse.ldif** file.

4.3.3. Restoring All Databases

The following procedures describe restoring all of the databases in the directory using the Directory Server Console and from the command line.

NOTE

Restoring a database from backup also restores the changelog.

IMPORTANT

While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Therefore, **stop all replication processes before attempting to restore a database**

4.3.3.1. Restoring All Databases from the Console

If the databases become corrupted, restore data from a previously generated backup using the Directory Server Console. This process consists of stopping the server and then copying the databases and associated index files from the backup location to the database directory.

WARNING

Restoring databases overwrites any existing database files.

IMPORTANT

While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Therefore, **stop all replication processes before attempting to restore a database**

To restore databases from a previously created backup:

1. In the Directory Server Console, select the **Tasks** tab.
2. Click **Restore Directory Server**.

3. Select the backup from the **Available Backups** list, or enter the full path to a valid backup in the **Directory** text box.

The **Available Backups** list shows all backups located in the default directory, **/var/lib/dirsrv/slapd-*instance_name*/bak/***backup_directory*. *backup_directory* is the directory of the most recent backup, in the form *serverID-YYYY_MM_DD_hhmmss*.

4.3.3.2. Restoring Databases from the Command Line

There are three ways to restore databases from the command line:

- Using the **bak2db** command-line script. This script requires the server to be shut down.
- Using the **bak2db.pl** Perl script. This script works while the server is running.

- Creating a temporary entry under **cn=restore,cn=tasks,cn=config**. This method can also be run while the server is running.

IMPORTANT

While restoring databases, the server must be running (with the exception of running the **bak2db** command-line script). However, the databases will be unavailable for processing operations during the restore.

Therefore, **stop all replication processes before attempting to restore a database**

4.3.3.2.1. Using the bak2db Command-Line Script

1. If the Directory Server is running, stop it:

```
[root@server ~]# service dirsrv stop instance
```

2. Run the **bak2db** command-line script. The **bak2db** script requires the full path and name of the input file.

```
[root@server ~]# /etc/dirsrv/slapd-instance_name/bak2db
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2017_04_30_11_48_30
```

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.

4.3.3.2.2. Using bak2db.pl Perl Script

Run the **bak2db.pl** Perl script.

```
[root@server ~]# /usr/lib[64]/dirsrv/slapd-example/bak2db.pl -D "cn=Directory Manager" -w secret -a
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2017_04_30_11_48_30
```

For more information on using this Perl script, see the *Directory Server Configuration and Command-Line Tool Reference*.

IMPORTANT

While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Therefore, **stop all replication processes before attempting to restore a database**

Option	Description
-a	Defines the full path and name of the input file.
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.

4.3.3.2.3. Restoring the Database through the cn=tasks Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=restore,cn=tasks,cn=config** to initiate a restore operation.

IMPORTANT

While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Therefore, **stop all replication processes before attempting to restore a database**

The restore task entry requires three attributes, the same as the backup task:

- A unique name (***cn***).
- The directory from which to retrieve the backup file (***nsArchiveDir***).
- The type of database (***nsDatabaseType***); the only option is **ldbm database**.

The entry is simply added using **ldapmodify**, as described in Section 3.2.4, "Adding and Modifying Entries Using ldapmodify". For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example restore,cn=restore,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example restore
nsArchiveDir: /export/backups/
nsDatabaseType: ldbm database
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on the available attributes for running database restore tasks under the **cn=tasks** entries.

4.3.4. Restoring a Single Database

It is possible to restore a single database through the command line, but not in the Directory Server Console. To restore a single database:

1. Stop the Directory Server if it is running.

   ```
   service dirsrv stop instance
   ```

2. Restore the back end from the **/var/lib/dirsrv/slapd-*instance_name*/bak** archives with the **bak2db** script, using the **-n** parameter to specify the database name. For example:

   ```
   bak2db /var/lib/dirsrv/slapd-instance_name/bak/backup_file -n userRoot
   ```

3. Restart the Directory Server.

   ```
   service dirsrv start instance
   ```

NOTE

If the Directory Server fails to start, remove the database transaction log files in **/var/lib/dirsrv/slapd-*instance_name*/db/log.###**, then retry starting the server.

4.3.5. Restoring Databases That Include Replicated Entries

Several situations can occur when a supplier server is restored:

- The consumer servers are also restored.

  If all databases are restored from backups taken at the same time (so that the data are in sync), the consumers remain synchronized with the supplier, and it is not necessary to do anything else. Replication resumes without interruption.

- Only the supplier is restored.

  If only the supplier is restored or if the consumers are restored from backups taken at a different times, reinitialize the consumers for the supplier to update the data in the database. If only the supplier is restored or if the consumers are restored from backups taken at a different times, reinitialize the consumers for the supplier to update the data in the database.

- Changelog entries have not yet expired on the supplier server.

  If the supplier's changelog has not expired since the database backup was taken, then restore the local consumer and continue with normal operations. This situation occurs only if the backup was taken within a period of time that is shorter than the value set for the maximum changelog age attribute, ***nsslapd-***

changelogmaxage, in the **cn=changelog5,cn=config** entry. For more information about this option, see the *Directory Server Configuration and Command-Line Tool Reference* .

Directory Server automatically detects the compatibility between the replica and its changelog. If a mismatch is detected, the server removes the old changelog file and creates a new, empty one.

- Changelog entries have expired on the supplier server since the time of the local backup.

  If changelog entries have expired, reinitialize the consumer. For more information on reinitializing consumers, see Section 11.15, "Initializing Consumers".

The changelog associated with the restored database will be erased during the restore operation. A message will be logged to the supplier servers' log files indicating that reinitialization is required.

For information on managing replication, see Chapter 11, *Managing Replication*.

4.3.6. Restoring the dse.ldif Configuration File

The directory creates two backup copies of the **dse.ldif** file in the **/etc/dirsrv/slapd-*instance_name*** directory. The **dse.ldif.startOK** file records a copy of the **dse.ldif** file at server start up. The **dse.ldif.bak** file contains a backup of the most recent changes to the **dse.ldif** file. Use the version with the most recent changes to restore the directory.

To restore the **dse.ldif** configuration file:

1. Stop the server.

   ```
   service dirsrv stop instance
   ```

2. Restore the database as outlined in Section 4.3.4, "Restoring a Single Database" to copy the backup copy of the **dse.ldif** file into the directory.

3. Restart the server.

   ```
   service dirsrv restart instance
   ```

CHAPTER 5. MANAGING ATTRIBUTES AND VALUES

Red Hat Directory Server provides several different mechanisms for dynamically and automatically maintaining some types of attributes on directory entries. These plug-ins and configuration options simplify managing directory data and expressing relationships between entries.

Part of the characteristic of entries are their *relationships* to each other. Obviously, a manager has an employee, so those two entries are related. Groups are associated with their members. There are less apparent relationships, too, like between entries which share a common physical location.

Red Hat Directory Server provides several different ways that these relationships between entries can be maintained smoothly and consistently. There are several plug-ins can apply or generate attributes automatically as part of the data within the directory, including classes of service, linking attributes, and generating unique numeric attribute values.

5.1. ENFORCING ATTRIBUTE UNIQUENESS

Attribute uniqueness means that the Directory Server requires that all new or edited attributes always have unique values. Attribute uniqueness is enforced through a plug-in. A new instance of the Attribute Uniqueness Plug-in must be created for every attribute for which values must be unique.

5.1.1. Attribute Uniqueness Plug-in Syntax

Configuration information for the Attribute Uniqueness Plug-in is specified in an entry under **cn=plugins,cn=config** entry. There are two possible syntaxes for ***nsslapd-pluginarg*** attributes.

NOTE

To enforce uniqueness of another attribute than the ones in these example, copy and paste the default Attribute Uniqueness Plug-in entry, and being care to change only the attributes described here.

Use the following syntax to perform the uniqueness check under a suffix or subtree:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
...
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute_name
nsslapd-pluginarg1: dn1
nsslapd-pluginarg2: dn2
...
```

- Any value can be given to the ***cn*** attribute to name the plug-in. The name should be descriptive.
- The ***cn*** attribute does not contain the name of the attribute which is checked for uniqueness.
- Only one attribute can be specified on which the uniqueness check will be performed.
- It is possible to specify several DNs of suffixes or subtrees in which to perform the uniqueness check by incrementing the ***nsslapd-pluginarg*** attribute suffix by one each time.

The variable components of the Attribute Uniqueness Plug-in syntax are described in Table 5.1, "Attribute Uniqueness Plug-in Variables".

Use the following syntax to specify to perform the uniqueness check below an entry containing a specified object class:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
...
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute=attribute_name
nsslapd-pluginarg1: markerObjectClass=objectclass1
nsslapd-pluginarg2: requiredObjectClass=objectclass2
...
```

- Any value can be given to the ***cn*** attribute to name the plug-in. The name should be descriptive.
- The ***cn*** attribute does not contain the name of the attribute which is checked for uniqueness.

- Only one attribute can be specified on which the uniqueness check will be performed.
- If the ***nsslapd-pluginarg0*** attribute begins with **attribute=***attribute_name*, then the server expects the ***nsslapd-pluginarg1*** attribute to include a **markerObjectClass** value.

The variable components of the attribute uniqueness plug-in syntax are described in Table 5.1, "Attribute Uniqueness Plug-in Variables".

Table 5.1. Attribute Uniqueness Plug-in Variables

Variable	Definition
descriptive_plugin_name	Specifies the name of this instance of the Attribute Uniqueness Plug-in. It is not required that the name of the attribute for which to ensure uniqueness be included, but it is advisable. For example, **cn=mail uniqueness,cn=plugins,cn=config**.
state	Defines whether the plug-in is enabled or disabled. Acceptable values are **on** or **off**.
attribute_name	The name of the attribute for which to ensure unique values. Only one attribute can be named.
dn	The DN of the suffix or subtree in which to ensure attribute uniqueness. To specify several suffixes or subtrees, increment the suffix of the ***nsslapd-pluginarg*** attribute by one for each additional suffix or subtree.
attribute= *attribute_name*	The name of the attribute for which to ensure unique values. Only one attribute can be named.
markerObjectClass= *objectclass1*	Attribute uniqueness will be checked under the entry belonging to the DN of the updated entry that has the object class specified in the **markerObjectClass** keyword. Do not include a space before or after the equals sign.
requiredObjectClass= *objectclass2*	*Optional*. When using the **markerObjectClass** keyword to specify the scope of the uniqueness check instead of a DN, it is also possible to specify to perform the check only if the updated entry contains the objectclass specified in the **requiredObjectClass** keyword. Do not include a space before or after the equals sign.

5.1.2. Creating an Instance of the Attribute Uniqueness Plug-in

To ensure that a particular attribute in the directory always has unique values, create an instance of the Attribute Uniqueness Plug-in for the attribute to check. For example, to ensure that every entry in the directory that includes a ***mail*** attribute has a unique value for that attribute, create a mail uniqueness plug-in.

To create an instance of the Attribute Uniqueness Plug-in, modify the Directory Server configuration to add an entry for the new plug-in under the **cn=plugins,cn=config** entry. The format of the new entry must conform to the syntax described in Section 5.1.1, "Attribute Uniqueness Plug-in Syntax" .

NOTE

Red Hat strongly encourages you to copy and paste an existing Attribute Uniqueness Plug-in entry and only modify the attributes listed below.

For example, to create an instance the Attribute Uniqueness Plug-in for the mail attribute:

1. Stop the Directory Server. Changes to the **dse.ldif** file are not saved if it is edited while the server is running.

```
service dirsrv stop instance_name
```

2. In the **dse.ldif** file, locate the entry for the Attribute Uniqueness Plug-in, **cn=attribute uniqueness,cn=plugins,cn=config**.
3. Copy the entire entry. The entry ends in an empty line; copy the empty line, too.
4. Paste the copied Attribute Uniqueness Plug-in entry at the end of the file.
5. Modify the Attribute Uniqueness Plug-in entry attributes for the new attribute information:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=example,dc=com
...
```

6. Restart the Directory Server.

```
service dirsrv start instance_name
```

In this example, the uniqueness check will be performed on every entry in the **dc=example,dc=com** entry that includes the **mail** attribute.

5.1.3. Configuring Attribute Uniqueness

This section explains how to use Directory Server Console to view the plug-ins configured for the directory and how to modify the configuration of the Attribute Uniqueness Plug-ins.

5.1.3.1. Configuring Attribute Uniqueness Plug-ins from the Directory Server Console

1. In the Directory Server Console, select the **Configuration** tab; then, in the navigation tree, expand the **Plug-ins** folder, and select the Attribute Uniqueness Plug-in to modify.

   The configuration parameters for the plug-in are displayed in the right pane.

2. To add a suffix or subtree, click **Add**, and type a DN in the blank text field.

   To delete a suffix from the list, place the cursor in the text field to delete, and click **Delete**.

   To avoid using a DN, enter the **markerObjectClass** keyword. With this syntax, it is possible to click **Add** again to specify a **requiredObjectClass**, as described in Section 5.1.1, “Attribute Uniqueness Plug-in Syntax”.

   NOTE

   Do *not* add an attribute name to the list. To check the uniqueness of other attributes, create a new instance of the Attribute Uniqueness Plug-in for the attribute to check. For information, see Section 5.1.2, “Creating an Instance of the Attribute Uniqueness Plug-in”.

3. Click **Save**.

5.1.3.2. Configuring Attribute Uniqueness Plug-ins from the Command Line

This section provides information about configuring the plug-in from the command line.

- Section 5.1.3.2.1, “Specifying a Suffix or Subtree”
- Section 5.1.3.2.2, “Using the markerObjectClass and requiredObjectClass Keywords”

5.1.3.2.1. Specifying a Suffix or Subtree

The suffix or subtrees which the plug-in checks to ensure attribute uniqueness are defined using the ***nsslapd-pluginarg*** attribute in the entry defining the plug-in.

To specify the subtree or subtrees, use **ldapmodify** to send LDIF update statements. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=mail uniqueness,cn=plugins,cn=config
```

```
changetype: modify
add: nsslapd-pluginarg2 nsslapd-pluginarg3
nsslapd-pluginarg2: ou=Engineering,dc=example,dc=com
nsslapd-pluginarg3: ou=Sales,dc=example,dc=com
```

This example LDIF statement modified the Attribute Uniqueness Plug-in to check the uniqueness of the ***mail*** attribute under the subtrees **dc=example,dc=com**, **ou=Engineering,dc=example,dc=com**, and **ou=Sales,dc=example,dc=com**.

Whenever this type of configuration change is made, restart the server.

```
service dirsrv restart instance_name
```

For information on restarting the server, see Section 1.3, “Starting and Stopping Servers”.

5.1.3.2.2. Using the markerObjectClass and requiredObjectClass Keywords

Instead of specifying a suffix or subtree in the configuration of an Attribute Uniqueness Plug-in, perform the check under the entry belonging to the DN of the updated entry that has the object class given in the **markerObjectClass** keyword.

To specify to perform the uniqueness check under the entry in the DN of the updated entry that contains the organizational unit (**ou**) object class, copy and paste an existing Attribute Uniqueness Plug-in entry, and change the following attributes:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=organizationalUnit
...
```

If the server should not check every entry in the organization unit, limit the scope by setting the check to be performed only if the updated entry contains a specified object class.

For example, if the uniqueness of the ***mail*** attribute is checked, it is probably only necessary to perform the check when adding or modifying entries with the ***person*** or ***inetorgperson*** object class.

Restrict the scope of the check by using the **requiredObjectClass** keyword, as shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=organizationalUnit
nsslapd-pluginarg2: requiredObjectClass=person
...
```

The **markerObjectClass** or **requiredObjectClass** keywords *cannot* be repeated by incrementing the counter in the **nsslapd-pluginarg** attribute suffix. These keywords can only be used once per Attribute Uniqueness Plug-in instance.

NOTE

The ***nsslapd-pluginarg0*** attribute always contains the name of the attribute for which to ensure uniqueness.

5.1.4. Attribute Uniqueness Plug-in Syntax Examples

This section contains examples of Attribute Uniqueness Plug-in syntax in the **dse.ldif** file.

- Section 5.1.4.1, “Specifying One Attribute and One Subtree”
- Section 5.1.4.2, “Specifying One Attribute and Multiple Subtrees”

5.1.4.1. Specifying One Attribute and One Subtree

This example configures the plug-in to ensure the uniqueness of the **mail** attribute under the **dc=example,dc=com** subtree.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=example,dc=com
...
```

5.1.4.2. Specifying One Attribute and Multiple Subtrees

It is possible use a single plug-in instance to check for the uniqueness of an attribute within multiple subtrees, which means that the attribute value must be unique *within* each subtree but not unique across all subtrees. This example configures the Attribute Uniqueness Plug-in to ensure the uniqueness of the ***mail*** attribute for separate subtrees, **l=Chicago,dc=example,dc=com** and **l=Boston,dc=example,dc=com**.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: l=Chicago,dc=example,dc=com
nsslapd-pluginarg2: l=Boston,dc=example,dc=com
...
```

NOTE

The ***nsslapd-pluginarg0*** attribute always contains the name of the attribute for which to ensure uniqueness. All other occurrences of the ***nsslapd-pluginarg***, such as ***nsslapd-pluginarg1***, contain DNs.

With this configuration, the plug-in allows an instance of a value for the ***mail*** attribute to exist once under the **l=Chicago,dc=example,dc=com** subtree and once under the **l=Boston,dc=example,dc=com** subtree. For example, the following two attribute-value settings are allowed:

```
mail=bjensen,l=Chicago,dc=example,dc=com
mail=bjensen,l=Boston,dc=example,dc=com
```

To ensure that only one instance of a value exists under both subtrees, configure the plug-in to ensure uniqueness for the entire **dc=example,dc=com** subtree.

5.2. ASSIGNING CLASS OF SERVICE

A *class of service definition* (CoS) shares attributes between entries in a way that is transparent to applications. CoS simplifies entry management and reduces storage requirements.

Clients of the Directory Server read the attributes in a user's entry. With CoS, some attribute values may not be stored within the entry itself. Instead, these attribute values are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of two types of entry in the directory:

- *CoS definition entry.* The CoS definition entry identifies the type of CoS used. Like the role definition entry, it inherits from the **LDAPsubentry** object class. The CoS definition entry is below the branch at which it is effective.
- *Template entry.* The CoS template entry contains a list of the shared attribute values. Changes to the template entry attribute values are automatically applied to all the entries within the scope of the CoS. A single CoS might have more than one template entry associated with it.

The CoS definition entry and template entry interact to provide attribute information to their target entries, any entry within the scope of the CoS.

5.2.1. About the CoS Definition Entry

The CoS definition entry is an instance of the **cosSuperDefinition** object class. The CoS definition entry also contains one of three object class that specifies the type of template entry it uses to generate the entry. The target entries which interact with the CoS share the same parent as the CoS definition entry.

There are three types of CoS, defined using three types of CoS definition entries:

- *Pointer CoS.* A pointer CoS identifies the template entry using the template DN only.
- *Indirect CoS.* An indirect CoS identifies the template entry using the value of one of the target entry's attributes. For example, an indirect CoS might specify the ***manager*** attribute of a target entry. The value of the ***manager*** attribute is then used to identify the template entry.

  The target entry's attribute must be single-valued and contain a DN.
- *Classic CoS.* A classic CoS identifies the template entry using a combination of the template entry's base DN and the value of one of the target entry's attributes.

For more information about the object classes and attributes associated with each type of CoS, see Section 5.2.11, "Managing CoS from the Command Line".

If the CoS logic detects that an entry contains an attribute for which the CoS is generating values, the CoS, by default, supplies the client application with the attribute value in the entry itself. However, the CoS definition entry can control this behavior.

5.2.2. About the CoS Template Entry

The CoS template entry contains the value or values of the attributes generated by the CoS logic. The CoS template entry contains a general object class of **cosTemplate**. The CoS template entries for a given CoS are stored in the directory tree along with the CoS definition.

The relative distinguished name (RDN) of the template entry is determined by one of the following:

- The DN of the template entry alone. This type of template is associated with a pointer CoS definition.
- The value of one of the target entry's attributes. The attribute used to provide the relative DN to the template entry is specified in the CoS definition entry using the ***cosIndirectSpecifier*** attribute. This type of template is associated with an indirect CoS definition.
- By a combination of the DN of the subtree where the CoS performs a one level search for templates and the value of one of the target entry's attributes. This type of template is associated with a classic CoS definition.

5.2.3. How a Pointer CoS Works

An administrator creates a pointer CoS that shares a common postal code with all of the entries stored under **dc=example,dc=com**. The three entries for this CoS appear as illustrated in Figure 5.1, "Sample Pointer CoS".

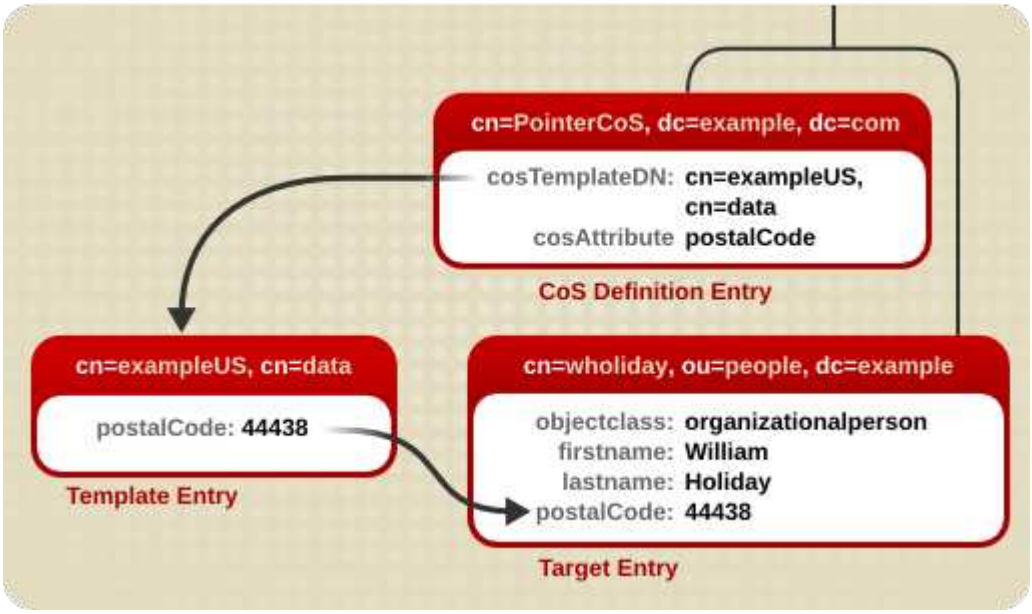


Figure 5.1. Sample Pointer CoS

In this example, the template entry is identified by its DN, **cn=exampleUS,cn=data**, in the CoS definition entry. Each time the ***postalCode*** attribute is queried on the entry **cn=wholiday,ou=people,dc=example,dc=com**, the Directory Server returns the value available in the template entry **cn=exampleUS,cn=data**.

5.2.4. How an Indirect CoS Works

An administrator creates an indirect CoS that uses the ***manager*** attribute of the target entry to identify the template entry. The three CoS entries appear as illustrated in Figure 5.2, "Sample Indirect CoS".

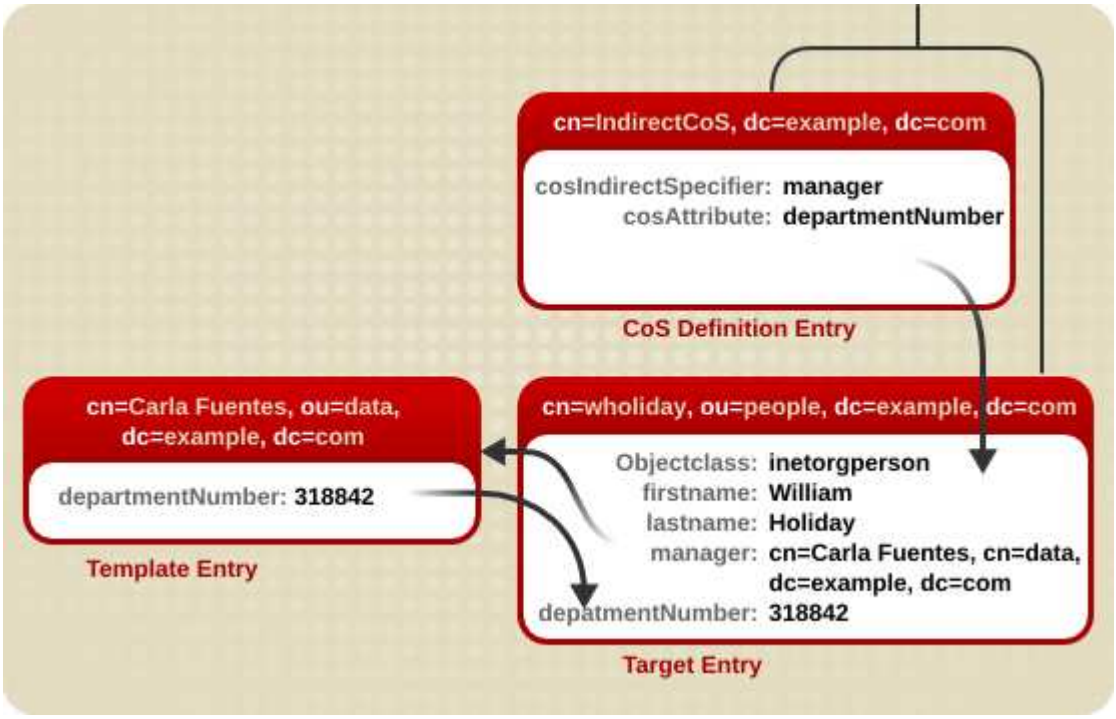


Figure 5.2. Sample Indirect CoS

In this example, the target entry for William Holiday contains the indirect specifier, the ***manager*** attribute. William's manager is Carla Fuentes, so the ***manager*** attribute contains a pointer to the DN of the template entry, **cn=Carla Fuentes,ou=people,dc=example,dc=com**. The template entry in turn provides the ***departmentNumber*** attribute value of **318842**.

5.2.5. How a Classic CoS Works

An administrator creates a classic CoS that uses a combination of the template DN and a CoS specifier to identify the template entry containing the postal code. The three CoS entries appear as illustrated in Figure 5.3, "Sample Classic CoS":

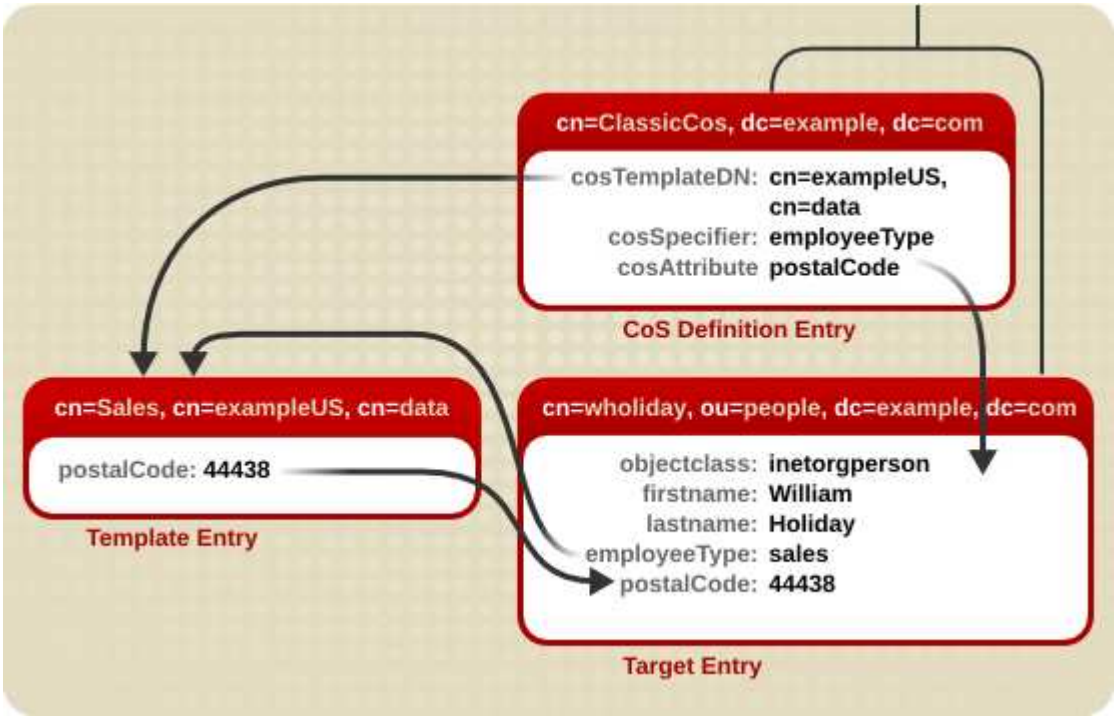


Figure 5.3. Sample Classic CoS

In this example, the CoS definition entry's ***cosSpecifier*** attribute specifies the ***employeeType*** attribute. This attribute, in combination with the template DN, identify the template entry as **cn=sales,cn=exampleUS,cn=data**. The template entry then provides the value of the ***postalCode*** attribute to the target entry.

5.2.6. Handling Physical Attribute Values

The ***cosAttribute*** attribute contains the name of another attribute which is governed by the class of service. This attribute allows an *override* qualifier (after the attribute value) which sets how the CoS handles existing attribute values on entries when it generates attribute values.

```
cosAttribute: attribute_name override
```

There are four *override* qualifiers.

Override Qualifier	Description
default	Only returns a generated value if there is no corresponding attribute value stored with the entry.
override	Always returns the value generated by the CoS, even when there is a value stored with the entry.
operational	Returns a generated attribute only if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order to be returned. When **operational** is used, it also overrides any existing attribute values. **NOTE** An attribute can only be made operational if it is defined as operational in the schema. For example, if the CoS generates a value for the ***description*** attribute, it is not possible to use the **operational** qualifier because this attribute is not marked operational in the schema.
operational-default	Only returns a generated value if there is no corresponding attribute value stored with the entry and if it is explicitly requested in the search.

If no qualifier is set, **default** is assumed.

For example, this pointer CoS definition entry indicates that it is associated with a template entry, **cn=exampleUS,ou=data,dc=example,dc=com**, that generates the value of the ***postalCode*** attribute. The **override** qualifier indicates that this value will take precedence over the value stored by the entries for the ***postalCode*** attribute:

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode override
```

NOTE

If an entry contains an attribute value generated by a CoS, the value of the attribute *cannot* be manually updated if it is defined with the operational or override qualifiers.

For more information about the CoS attributes, see the *Directory Server Configuration and Command-Line Tool Reference*.

5.2.7. Handling Multi-valued Attributes with CoS

Any attribute can be generated using a class of service – including multi-valued attributes. That introduces the potential for confusion. Which CoS supplies a value? Any of them or all of them? How is the value selected from competing CoS templates? Does the generated attribute use a single value or multiple values?

There are two ways to resolve this:

- Creating a rule to merge multiple CoS-generated attributes into the target entry. This results in multiple values in the target entry.
- Setting a priority to select one CoS value out of competing CoS definitions. This generates one single value for the target entry.

NOTE

Indirect CoS do not support the ***cosPriority*** attribute.

The way that the CoS handles multiple values for a CoS attribute is defined in whether it uses a *merge-schemes* qualifier.

```
cosAttribute: attribute override merge-schemes
```

NOTE

The *merge-schemes* qualifier does not affect how the CoS handles physical attribute values or the *override* qualifier. If there are multiple competing CoS templates or definitions, then the same *merge-schemes* and *override* qualifiers have to be set on every ***cosAttribute*** for every competing CoS definition. Otherwise, one combination is chosen arbitrarily from all possible CoS definitions.

Using the *merge-schemes* qualifier tells the CoS that it will, or can, generate multiple values for the managed attribute. There are two possible scenarios for having a multi-valued CoS attribute:

- One CoS template entry contains multiple instances of the managed CoS attribute, resulting in multiple values on the target entry. For example:

```
dn: cn=server access template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: mail.example.com
accessTo: irc.example.com
```

NOTE

This method only works with classic CoS.

- Multiple CoS definitions may define a class of service for the same target attribute, so there are multiple template entries. For example:

```
dn: cn=mail template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: mail.example.com

dn: cn=chat template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: irc.example.com
```

However, it may be that even if there are multiple CoS definitions, only one value should be generated for the attribute. If there are multiple CoS definitions, then the value is chosen arbitrarily. This is an unpredictable and unwieldy option. The way to control which CoS template to use is to set a ranking on the template – a *priority* – and the highest prioritized CoS always "wins" and provides the value.

It is fairly common for there to be multiple templates completing to provide a value. For example, there can be a multi-valued ***cosSpecifier*** attribute in the CoS definition entry. The template priority is set using the ***cosPriority*** attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

For example, a CoS template entry for generating a department number appears as follows:

```
dn: cn=data,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
departmentNumber: 71776
cosPriority: 0
```

This template entry contains the value for the ***departmentNumber*** attribute. It has a priority of zero, meaning this template takes precedence over any other conflicting templates that define a different ***departmentNumber*** value.

Templates that contain no ***cosPriority*** attribute are considered the lowest priority. Where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.

NOTE

The behavior for negative ***cosPriority*** values is not defined in Directory Server; do not enter negative values.

5.2.8. Searches for CoS-Specified Attributes

CoS definitions provide values for attributes in entries. For example, a CoS can set the ***postalCode*** attribute for every entry in a subtree. Searches against those CoS-defined attributes, however, do not behave like searches against regular entries.

If the CoS-defined attribute is indexed with any kind of index (including presence), then any attribute with a value set by the CoS is not returned with a search. For example:

- The ***postalCode*** attribute for Ted Morris is defined by a CoS.
- The ***postalCode*** attribute for Barbara Jensen is set in her entry.
- The ***postalCode*** attribute is indexed.

If an **ldapsearch** command uses the filter **(postalCode=*)**, then Barbara Jensen's entry is returned, while Ted Morris's is not.

If the CoS-defined attribute is *not* indexed, then every matching entry is returned in a search, regardless of whether the attribute value is set locally or with CoS. For example:

- The ***postalCode*** attribute for Ted Morris is defined by a CoS.
- The ***postalCode*** attribute for Barbara Jensen is set in her entry.
- The ***postalCode*** attribute is *not* indexed.

If an **ldapsearch** command uses the filter **(postalCode=*)**, then both Barbara Jensen's and Ted Morris's entries are returned.

CoS allows for an *override*, an identifier given to the ***cosAttribute*** attribute in the CoS entry, which means that local values for an attribute can override the CoS value. If an override is set on the CoS, then an **ldapsearch** operation will return a value for an entry even if the attribute is indexed, as long as there is a local value for the entry. Other entries which possess the CoS but do not have a local value will still not be returned in the **ldapsearch** operation.

Because of the potential issues with running LDAP search requests on CoS-defined attributes, take care when deciding which attributes to generate using a CoS.

5.2.9. Access Control and CoS

The server controls access to attributes generated by a CoS in exactly the same way as regular stored attributes. However, access control rules depending upon the value of attributes generated by CoS will not work. This is the same restriction that applies to using CoS-generated attributes in search filters.

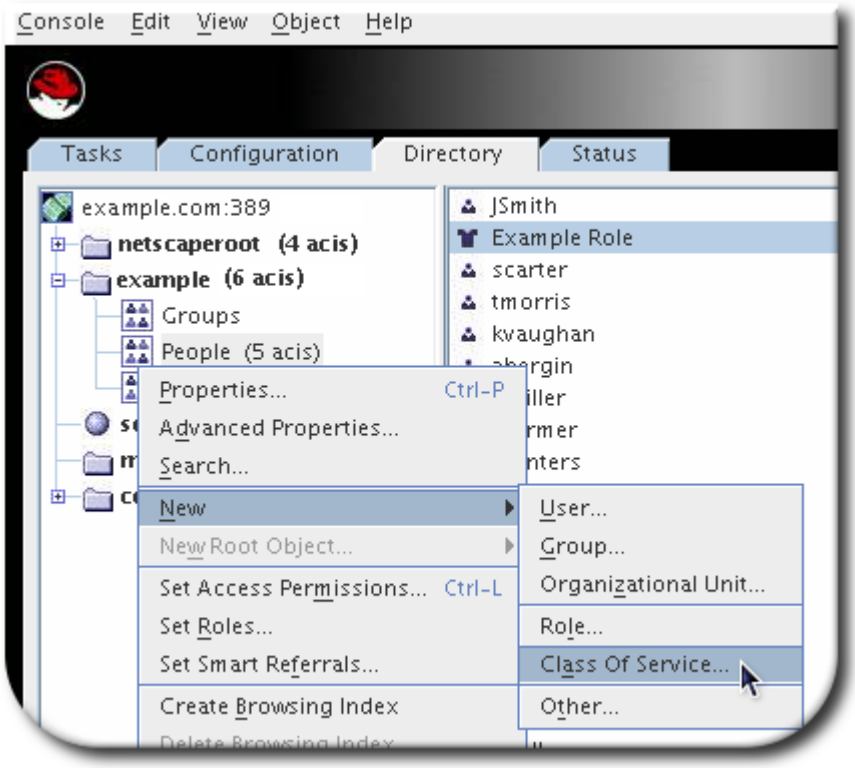
5.2.10. Managing CoS Using the Console

This section describes creating and editing CoS through the Directory Server Console:

- Section 5.2.10.1, "Creating a New CoS"
- Section 5.2.10.2, "Creating the CoS Template Entry"

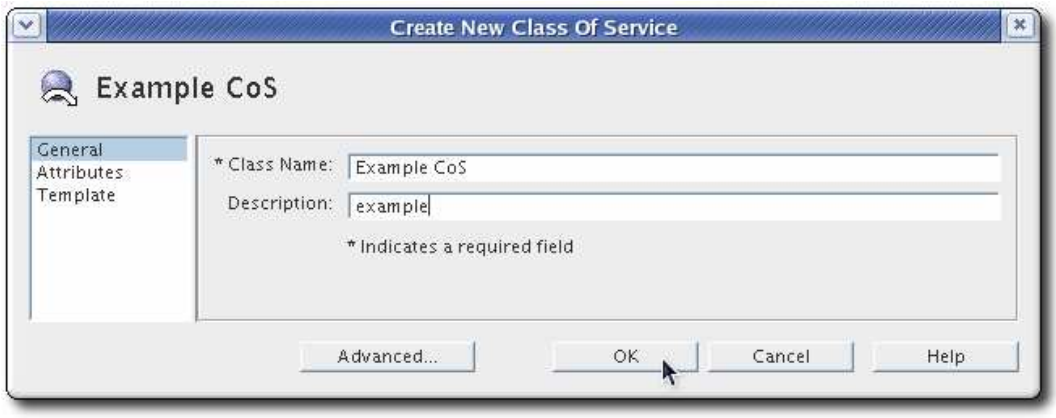
5.2.10.1. Creating a New CoS

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new class of service.
3. Go to the **Object** menu, and select **New > Class of Service**.



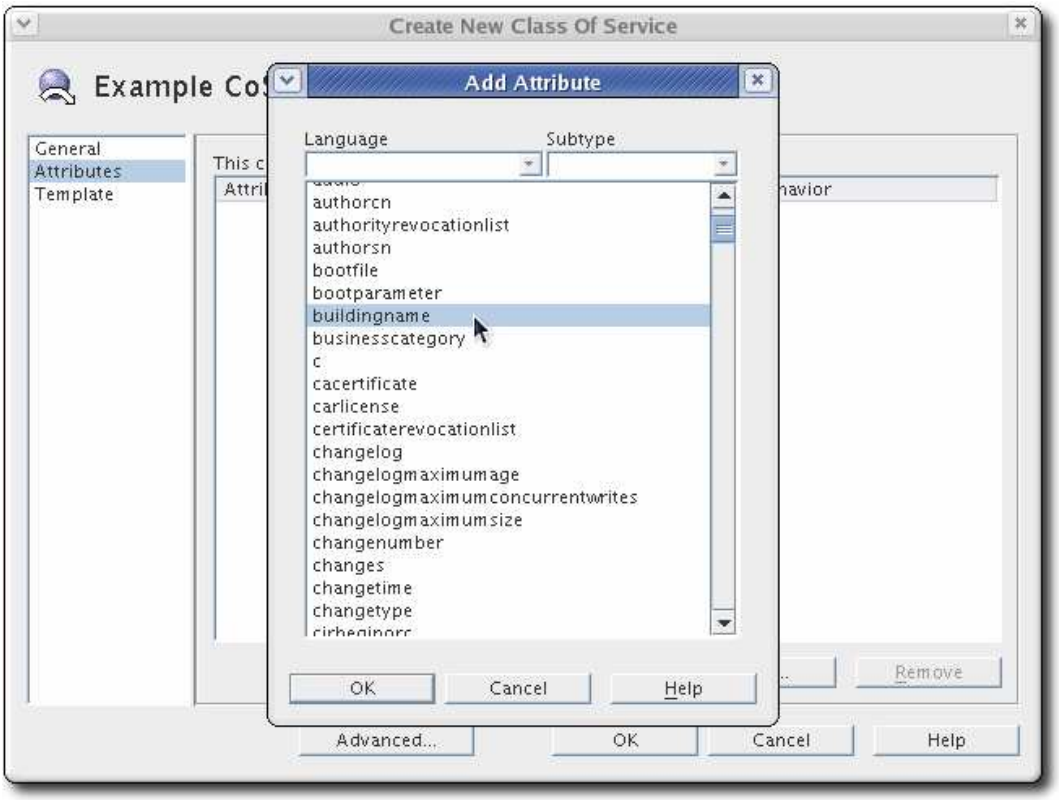
Alternatively, right-click the entry and select **New > Class of Service**.

4. Select **General** in the left pane. In the right pane, enter the name of the new class of service in the **Class Name** field. Enter a description of the class in the **Description** field.

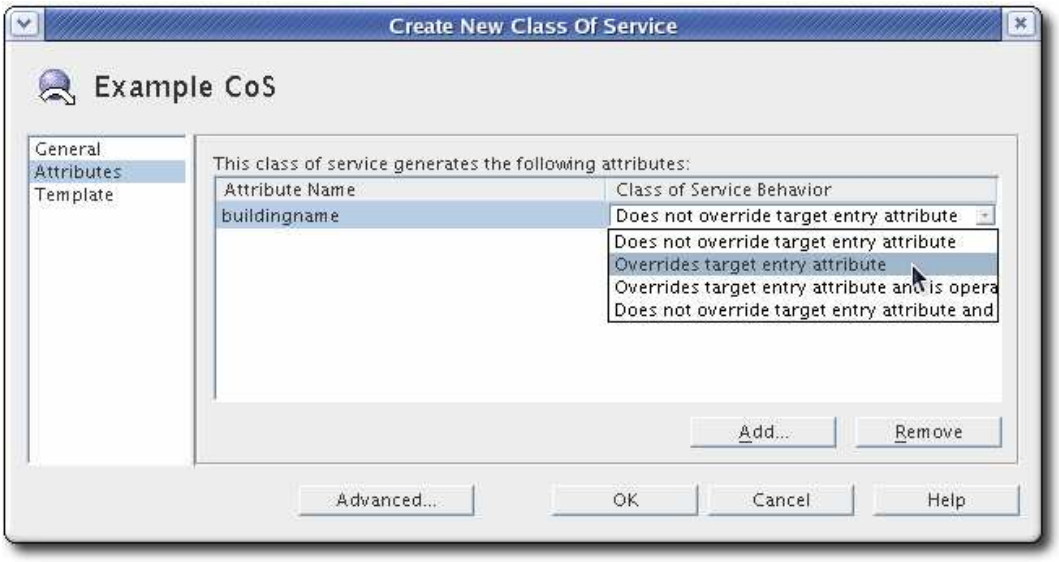


5. Click **Attributes** in the left pane. The right pane displays a list of attributes generated on the target entries.

   Click **Add** to browse the list of possible attributes and add them to the list.



6. After an attribute is added to the list, a drop-down list appears in the **Class of Service Behavior** column.

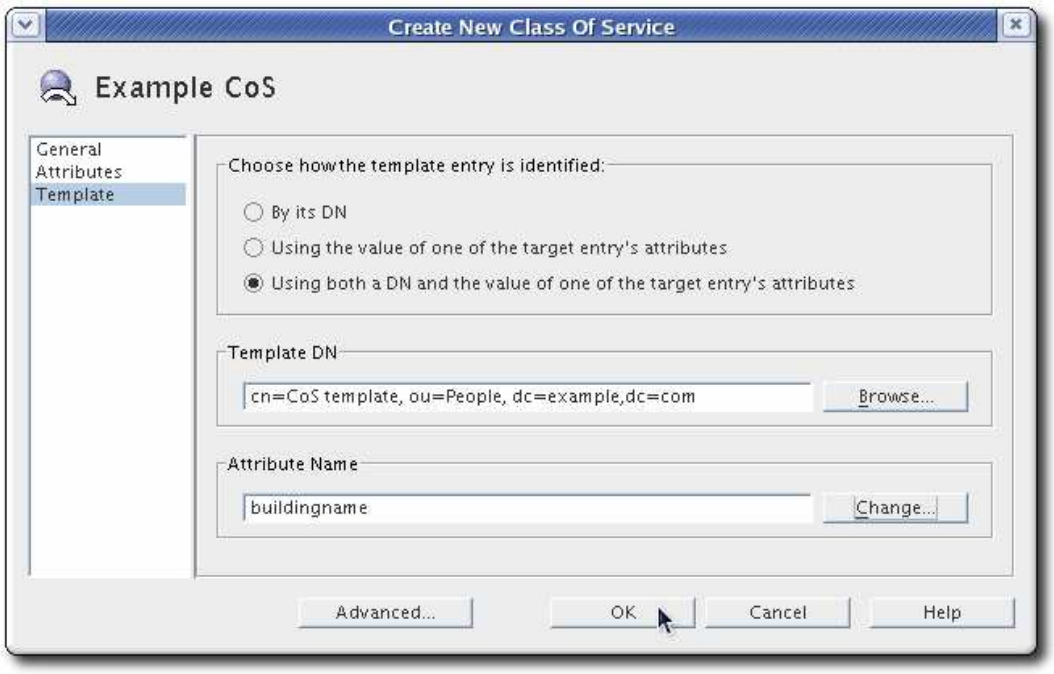


- Select **Does not override target entry attribute** to tell the directory to only return a generated value if there is no corresponding attribute value stored with the entry.
- Select **Overrides target entry attribute** to make the value of the attribute generated by the CoS override the local value.
- Select **Overrides target entry attribute and is operational** to make the attribute override the local value and to make the attribute operational, so that it is not visible to client applications unless explicitly requested.
- Select **Does not override target entry attribute and is operational** to tell the directory to return a generated value only if there is no corresponding attribute value stored with the entry and to make the attribute operational (so that it is not visible to client applications unless explicitly requested).

NOTE

An attribute can only be made operational if it is also defined as operational in the schema. For example, if a CoS generates a value for the ***description*** attribute, you cannot select **Overrides target entry attribute and is operational** because this attribute is not marked operational in the schema.

7. Click **Template** in the left pane. In the right pane, select how the template entry is identified.



   - *By its DN.* To have the template entry identified by only its DN (a pointer CoS), enter the DN of the template in the **Template DN** field. Click **Browse** to locate the DN on the local server. This will be an exact DN, such as **cn=CoS template,ou=People,dc=example,dc=com**.

   - *Using the value of one of the target entry's attribute.* To have the template entry identified by the value of one of the target entry's attributes (an indirect CoS), enter the attribute name in the **Attribute Name** field. Click **Change** to select a different attribute from the list of available attributes.

   - *Using both its DN and the value of one of the target entry's attributes.* To have the template entry identified by both its DN and the value of one of the target entry's attributes (a classic CoS), enter both a template DN and an attribute name. The template DN in a classic CoS is more general than for a pointer CoS; it references the suffix or subsuffix where the template entries will be. There can be more than one template for a classic CoS.

8. Click **OK**.

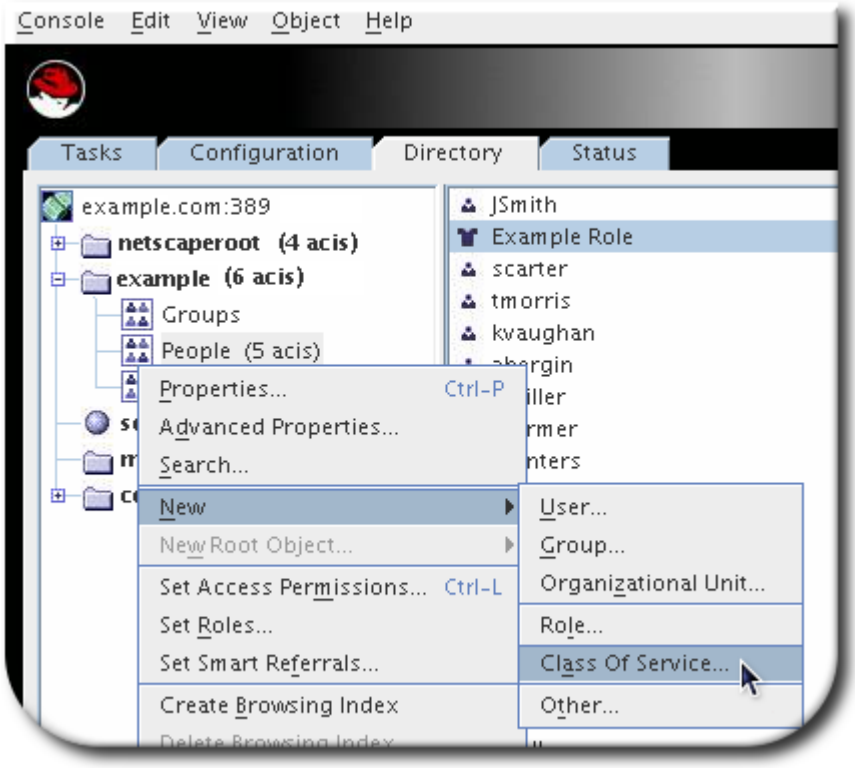
5.2.10.2. Creating the CoS Template Entry

For a pointer CoS or a classic CoS, there must be a template entry, according to the template DN set when the class of service was created. Although the template entries can be placed anywhere in the directory as long as the ***cosTemplateDn*** attribute reflects that DN, it is best to place the template entries under the CoS itself.

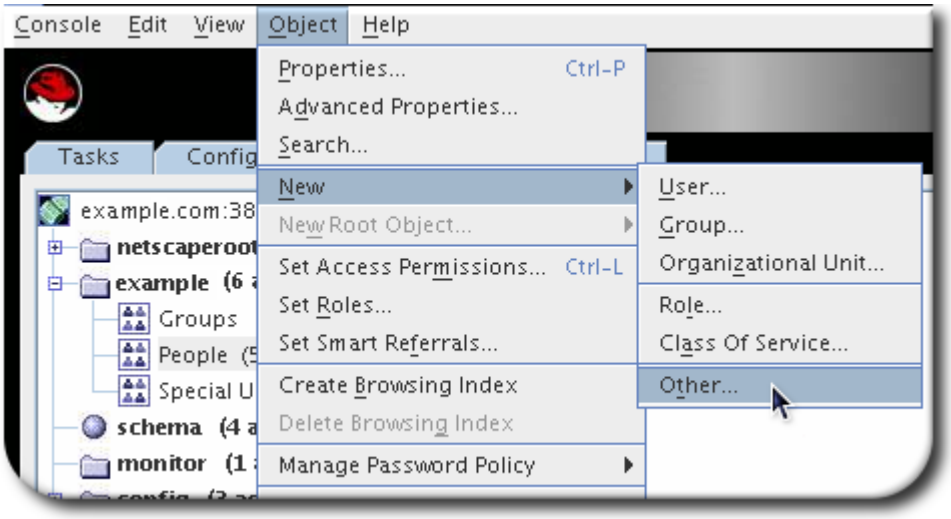
- For a pointer CoS, make sure that this entry reflects the exact DN given when the CoS was created.
- For a classic CoS, the template DN should be recursive, pointing back to the CoS entry itself as the base suffix for the template.

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry that contains the class of service.

   The CoS appears in the right pane with other entries.

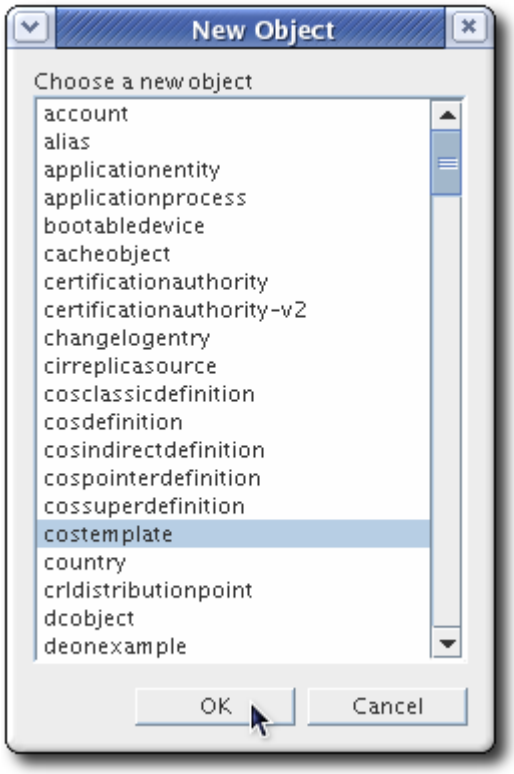


3. Right-click the CoS, and select **New > Other**.



Alternatively, select the CoS in the right pane, click **Object** in the menu at the top, and select **New > Other**.

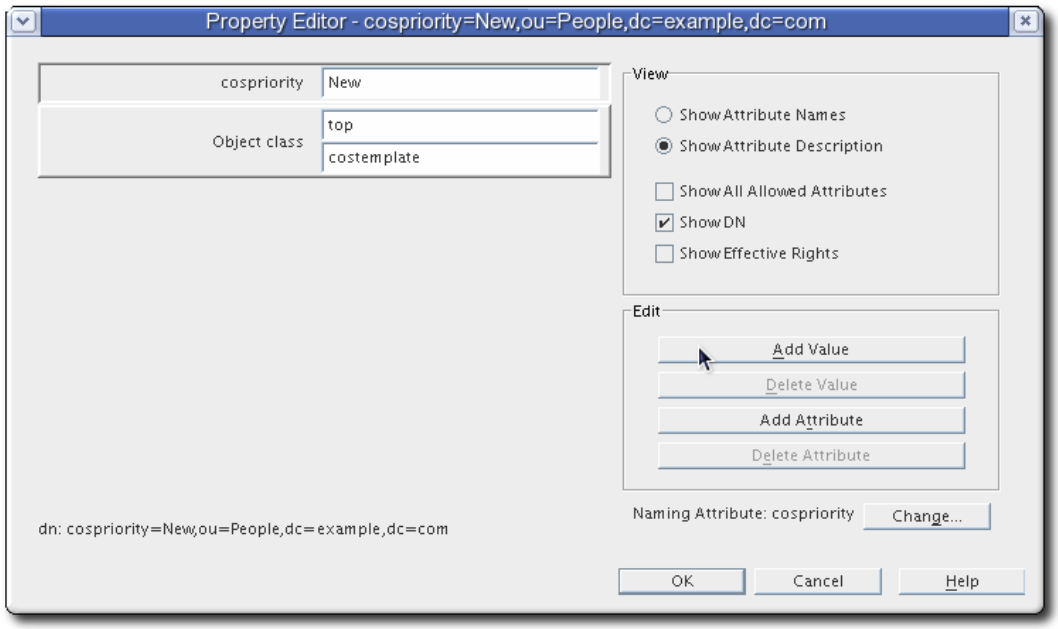
4. Select **cosTemplate** from the list of object classes.



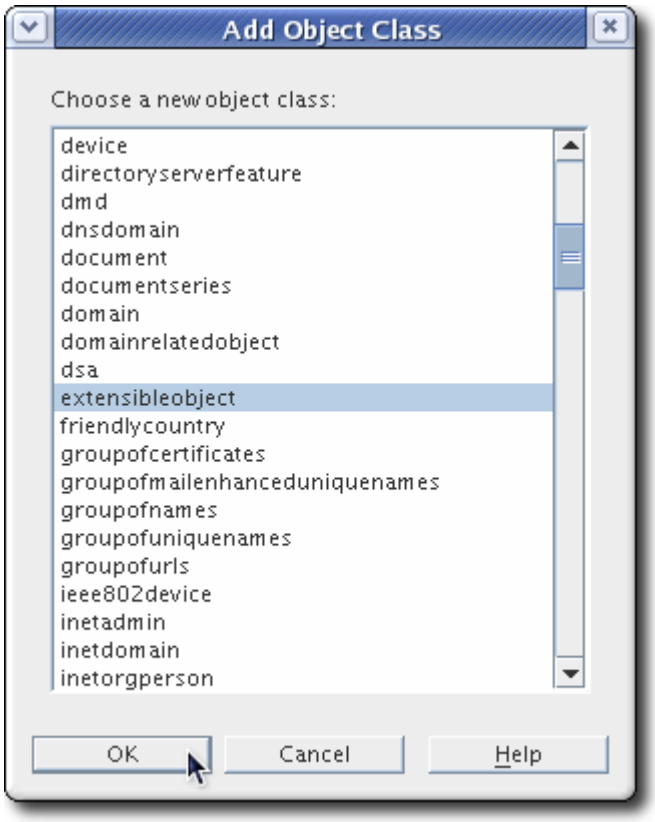
NOTE

The **LDAPsubentry** object class can be added to a new template entry. Making the CoS template entry an instance of the **LDAPsubentry** object class allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else (for example, if it is a user entry), the **LDAPsubentry** object class does not need to be added to the template entry.

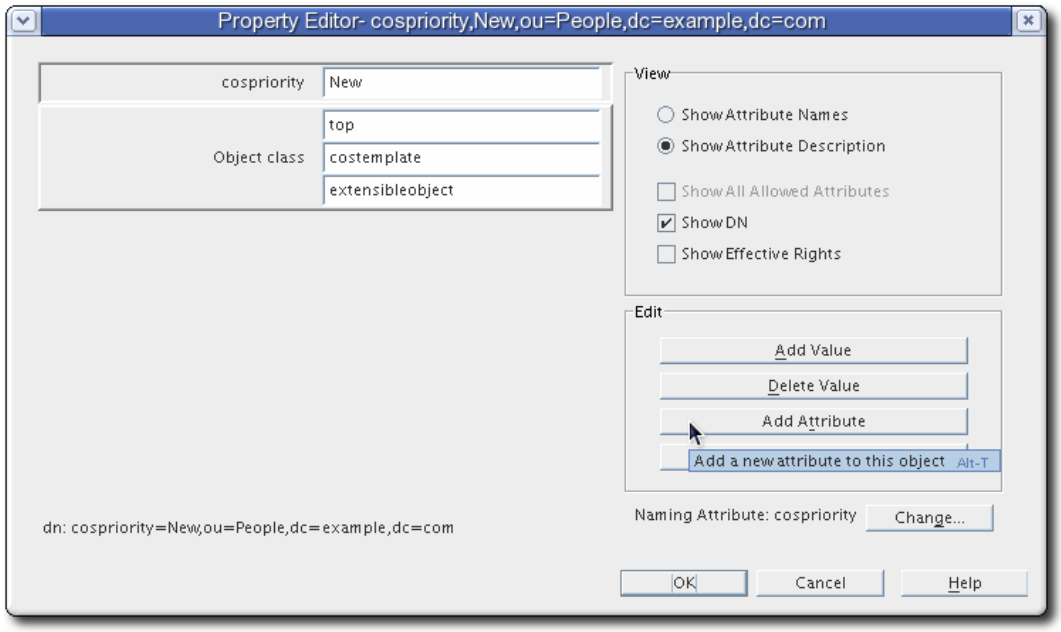
5. Select the object classes attribute, and click **Add Value**.



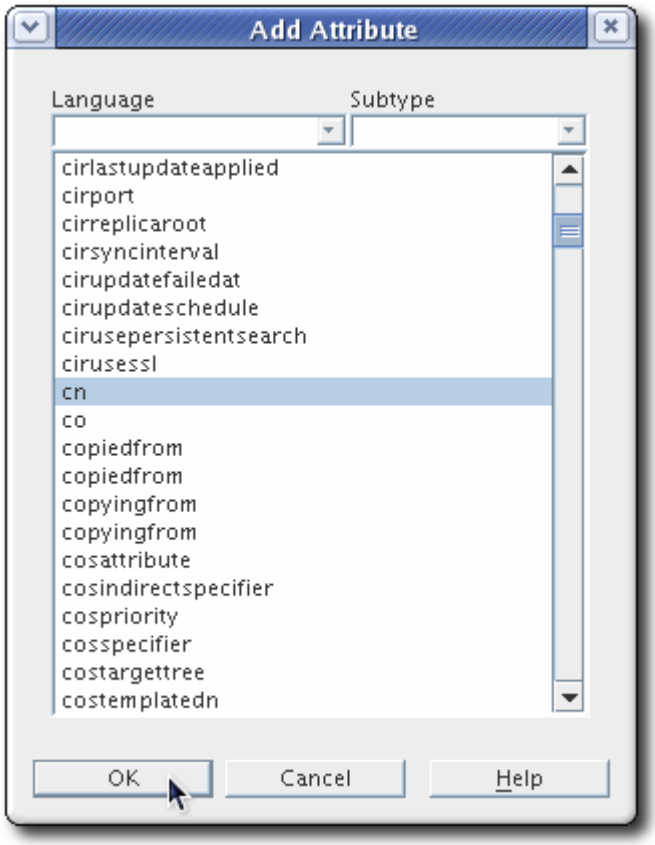
6. Add the **extensibleObject** object class. This makes it possible to add any attribute available in the directory.



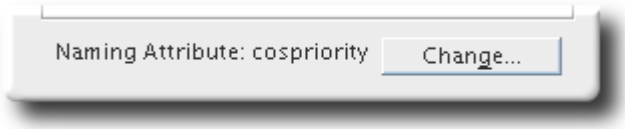
7. Click the **Add Attribute** button.



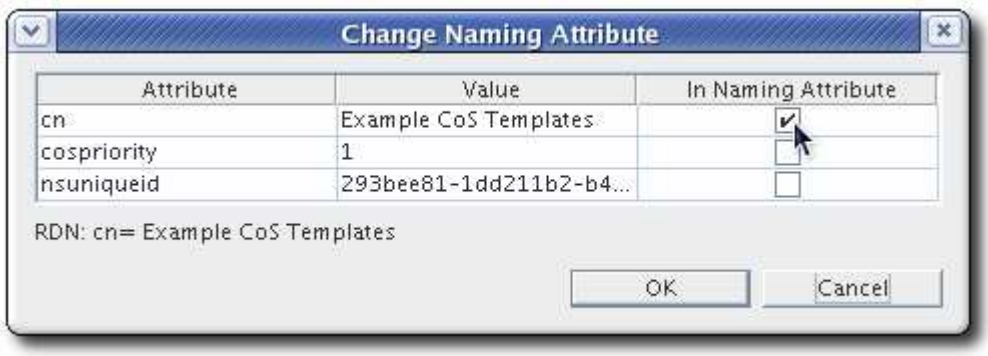
8. Add the ***cn*** attribute, and give it a value that corresponds to the attribute value in the target entry. For example, if the ***manager*** attribute is used to set the value for a classic CoS, give the ***cn*** a value of a manager's DN, such as **uid=bparker,ou=people,dc=example,dc=com**. Alternatively, set it to a role, such as **cn=QA Role,dc=example,dc=com** or a regular attribute value. For example, if the ***employeeType*** attribute is selected, it can be **full time** or **temporary**.



9. Click the **Change** button in the lower right corner to change the naming attribute.

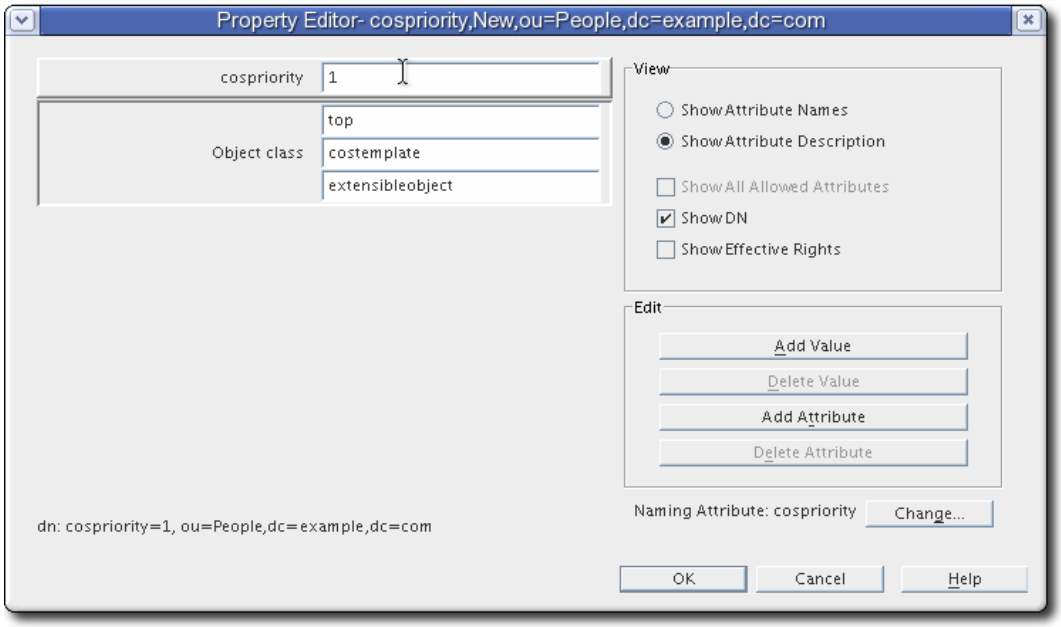


10. Use the ***cn*** of the entry as the naming attribute instead of ***cospriority***.



11. Click the **Add Attribute** button, and add the attributes listed in the CoS. The values used here will be used throughout the directory in the targeted entries.

12. Set the ***cospriority***. There may be more than one CoS that applies to a given attribute in an entry; the ***cospriority*** attribute ranks the importance of that particular CoS. The higher ***cospriority*** will take precedence in a conflict. The highest priority is **0**.



Templates that contain no ***cosPriority*** attribute are considered the lowest priority. In the case where two or more templates could supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.

NOTE

The behavior for negative ***cosPriority*** values is not defined in Directory Server; do not enter negative values.

NOTE

The ***cosPriority*** attribute is not supported by indirect CoS.

The CoS is visible in the left navigation pane once there are entries beneath it. For classic CoS, there can be multiple entries, according to the different potential values of the attribute specifier.

To edit the description or attributes generated on the target entry of an existing CoS, simply double-click the CoS entry listed in the **Directory** tab, and make the appropriate changes in the editor window.

5.2.11. Managing CoS from the Command Line

Because all configuration information and template data is stored as entries in the directory, standard LDAP tools can be used for CoS configuration and management.

- Section 5.2.11.1, "Creating the CoS Definition Entry from the Command Line"
- Section 5.2.11.2, "Creating the CoS Template Entry from the Command Line"
- Section 5.2.11.3, "Example of a Pointer CoS"
- Section 5.2.11.4, "Example of an Indirect CoS"
- Section 5.2.11.5, "Example of a Classic CoS"
- Section 5.2.11.6, "Searching for CoS Entries"

5.2.11.1. Creating the CoS Definition Entry from the Command Line

Each type of CoS requires a particular object class to be specified in the definition entry. All CoS definition object classes inherit from the **LDAPsubentry** object class and the **cosSuperDefinition** object class.

A pointer CoS uses the **cosPointerDefinition** object class. This object class identifies the template entry using an entry DN value specified in the ***cosTemplateDn*** attribute, as shown in Example 5.1, "An Example Pointer CoS Entry".

Example 5.1. An Example Pointer CoS Entry

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn:DN_string
cosAttribute:list_of_attributes qualifier
cn: pointerCoS
```

An indirect CoS uses the **cosIndirectDefinition** object class. This type of CoS identifies the template entry based on the value of one of the target entry's attributes, as specified in the ***cosIndirectSpecifier*** attribute. This is illustrated in Example 5.2, "An Example Indirect CoS Entry".

Example 5.2. An Example Indirect CoS Entry

```
dn: cn=indirectCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier:attribute_name
cosAttribute:list_of_attributes qualifier
cn: indirectCoS
```

A classic CoS uses the **cosClassicDefinition** object class. This identifies the template entry using both the template entry's DN (set in the ***cosTemplateDn*** attribute) and the value of one of the target entry's attributes (set in the ***cosSpecifier*** attribute). This is illustrated in Example 5.3, "An Example Classic CoS Entry".

Example 5.3. An Example Classic CoS Entry

```
dn: cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn:DN_string
cosSpecifier:attribute_name
cosAttribute:list_of_attributes qualifier
cn: classicCoS
```

For a class of service, the object class defines the type of CoS, and the supporting attributes identify which directory entries are affected by defining the CoS template. Every CoS has one additional attribute which can be defined for it: ***cosAttribute***. The purpose of a CoS is to supply attribute values across multiple entries; the ***cosAttribute*** attribute defines which attribute the CoS generates values for.

5.2.11.2. Creating the CoS Template Entry from the Command Line

Each template entry is an instance of the **cosTemplate** object class.

NOTE

Consider adding the **LDAPsubentry** object class to a new template entry. Making the CoS template entry an instance of the **LDAPsubentry** object classes allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else, such as a user entry, the **LDAPsubentry** object class does not need to be added to the template entry.

The CoS template entry also contains the attribute generated by the CoS (as specified in the ***cosAttribute*** attribute of the CoS definition entry) and the value for that attribute.

For example, a CoS template entry that provides a value for the ***postalCode*** attribute follows:

```
dn:cn=exampleUS,ou=data,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438
```

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

- Section 5.2.11.3, “Example of a Pointer CoS”
- Section 5.2.11.4, “Example of an Indirect CoS”
- Section 5.2.11.5, “Example of a Classic CoS”

5.2.11.3. Example of a Pointer CoS

Example Corporation's administrator is creating a pointer CoS that shares a common postal code with all entries in the **dc=example,dc=com** tree.

1. Add a new pointer CoS definition entry to the **dc=example,dc=com** suffix using **ldapmodify**:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Next, add the pointer CoS definition to the **dc=example,dc=com** root suffix.

```
dn: cn=pointerCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode
```

3. Create the template entry.

```
dn: cn=exampleUS,ou=data,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438
```

The CoS template entry (**cn=exampleUS,ou=data,dc=example,dc=com**) supplies the value stored in its ***postalCode*** attribute to any entries located under the **dc=example,dc=com** suffix. These entries are the target entries.

5.2.11.4. Example of an Indirect CoS

This indirect CoS uses the ***manager*** attribute of the target entry to identify the CoS template entry, which varies depending on the different values of the attribute.

1. Add a new indirect CoS definition entry to the **dc=example,dc=com** suffix:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=indirectCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

If the directory or modify the manager entries already contain the ***departmentNumber*** attribute, then no other attribute needs to be added to the manager entries. The definition entry looks in the target suffix (the entries under **dc=example,dc=com**) for entries containing the ***manager*** attribute because this attribute is specified in the ***cosIndirectSpecifier*** attribute of the definition entry). It then checks the ***departmentNumber*** value in the manager entry that is listed. The value of the ***departmentNumber*** attribute will automatically be relayed to all of the manager's subordinates that have the ***manager*** attribute. The value of ***departmentNumber*** will vary depending on the department number listed in the different manager's entries.

5.2.11.5. Example of a Classic CoS

The Example Corporation administrator is creating a classic CoS that automatically generates postal codes using a combination of the template DN and the attribute specified in the ***cosSpecifier*** attribute.

1. Add a new classic CoS definition entry to the **dc=example,dc=com** suffix.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=classicCoS,dc=example,dc=com
cosSpecifier: businessCategory
cosAttribute: postalCode override
```

2. Create the template entries for the sales and marketing departments. Add the CoS attributes to the template entry. The ***cn*** of the template sets the value of the ***businessCategory*** attribute in the target entry, and then the attributes are added or overwritten according to the value in the template:

```
dn: cn=sales,cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438

dn: cn=marketing,cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 99111
```

The classic CoS definition entry applies to all entries under the **dc=example,dc=com** suffix. Depending upon the combination of the ***businessCategory*** attribute found in the entry and the ***cosTemplateDn***, it can arrive at one of two templates. One, the sales template, provides a postal code specific to employees in the sales department. The marketing template provides a postal code specific to employees in the marketing department.

5.2.11.6. Searching for CoS Entries

CoS definition entries are *operational* entries and are not returned by default with regular searches. This means that if a CoS is defined under **ou=People,dc=example,dc=com**, for example, the following **ldapsearch** command will not return them:

```
ldapsearch -x -s sub -b ou=People,dc=example,dc=com “(objectclass=*)”
```

To return the CoS definition entries, add the **ldapSubEntry** object class to the CoS definition entries. For example:

```
dn: cn=pointerCoS,ou=People,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
objectclass: ldapSubEntry
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode override
```

Then use a special search filter, **(objectclass=ldapSubEntry)**, with the search. This filter can be added to any other search filter using OR (|):

```
ldapsearch -x -s sub -b ou=People,dc=example,dc=com “(|(objectclass=*)(objectclass=ldapSubEntry))”
```

This search returns all regular entries in addition to CoS definition entries in the **ou=People,dc=example,dc=com** subtree.

NOTE

The Console automatically shows CoS entries.

5.2.12. Creating Role-Based Attributes

Classic CoS schemes generate attribute values for an entry based on the role possessed by the entry. For example, role-based attributes can be used to set the server look-through limit on an entry-by-entry basis.

To create a role-based attribute, use the ***nsRole*** attribute as the ***cosSpecifier*** in the CoS definition entry of a classic CoS. Because the ***nsRole*** attribute can be multi-valued, CoS schemes can be defined that have more than one possible template entry. To resolve the ambiguity of which template entry to use, include the ***cosPriority*** attribute in the CoS template entry.

For example, this CoS allows members of the manager role to exceed the standard mailbox quota. The manager role entry is:

```
dn: cn=ManagerRole,ou=people,dc=example,dc=com
objectclass: top
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: ManagerRole
nsRoleFilter: ou=managers
Description: filtered role for managers
```

IMPORTANT

The ***nsRoleFilter*** attribute cannot accept virtual attribute values.

The classic CoS definition entry looks like:

```
dn: cn=managerCOS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=managerCOS,dc=example,dc=com
cosSpecifier: nsRole
cosAttribute: mailboxquota override
```

The ***cosTemplateDn*** attribute provides a value that, in combination with the attribute specified in the ***cosSpecifier*** attribute (in the example, the ***nsRole*** attribute of the target entry), identifies the CoS template entry. The CoS template entry provides the value for the ***mailboxquota*** attribute. An additional qualifier of **override** tells the CoS to override any existing ***mailboxquota*** attributes values in the target entry.

The corresponding CoS template entry looks as follows:

```
dn:cn=cn=ManagerRole\,ou=people\,dc=example\,dc=com,cn=managerCOS,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
mailboxquota: 1000000
```

The template provides the value for the ***mailboxquota*** attribute, **1000000**.

NOTE

The role entry and the CoS definition and template entries should be located at the same level in the directory tree.

5.3. LINKING ATTRIBUTES TO MANAGE ATTRIBUTE VALUES

A class of service dynamically supplies attribute values for entries which all have attributes with the *same value*, like building addresses, postal codes, or main office numbers. These are shared attribute values, which are updated in a single template entry.

Frequently, though, there are relationships between entries where there needs to be a way to express linkage between them, but the values (and possibly even the attributes) that express that relationship are different. Red Hat Directory Server provides a way to link specified attributes together, so that when one attribute in one entry is altered, a corresponding attribute on a related entry is automatically updated. (The link and managed attributes both have DN values. The value of the link attribute contains the DN of the entry for the plug-in to update; the managed attribute in the second entry has a DN value which points back to the original link entry.)

5.3.1. About Linking Attributes

The Linked Attributes Plug-in, allows multiple instances of the plug-in. Each instance configures one attribute which is manually maintained by the administrator (***linkType***) and one attribute which is automatically maintained by the plug-in (***managedType***).

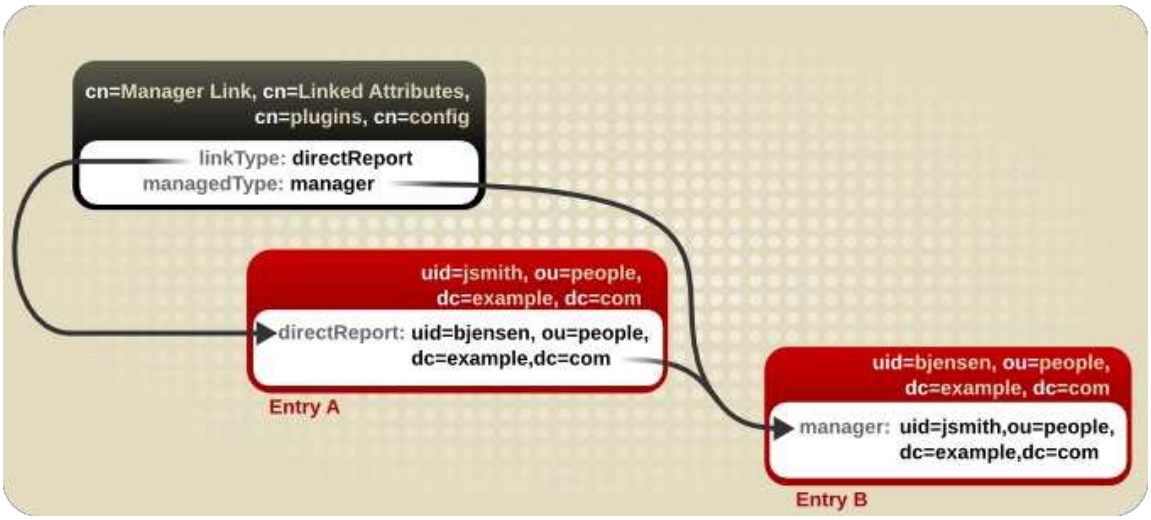


Figure 5.4. Basic Linked Attribute Configuration

NOTE

To preserve data consistency, only the plug-in process should maintain the managed attribute. Consider creating an ACI that will restrict all write access to any managed attribute. See Section 13.5.2, "Creating a New ACI" for information on setting ACIs.

A Linked Attribute Plug-in instance can be restricted to a single subtree within the directory. This can allow more flexible customization of attribute combinations and affected entries. If no scope is set, then the plug-in operates in the entire directory.

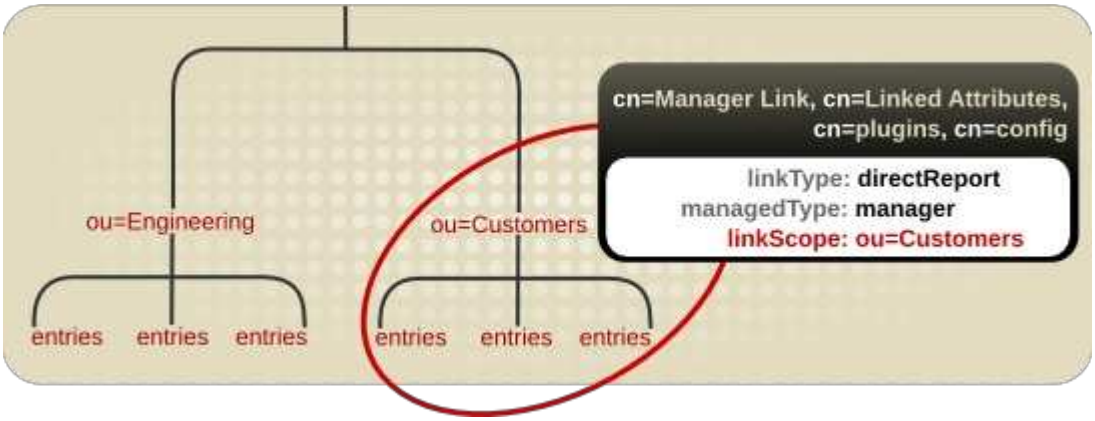


Figure 5.5. Restricting the Linked Attribute Plug-in to a Specific Subtree

When configuring the Linked Attribute Plug-in instance, certain configurations are required:

- Both the managed attribute and linked attribute must require the Distinguished Name syntax in their attribute definitions. The linked attributes are essentially managed cross-references, and the way that the plug-in handles these cross-references is by pulling the DN of the entry from the attribute value.

  For information on planning custom schema elements, see Chapter 8, *Managing the Directory Schema*.

- Each Linked Attribute Plug-in instance must be local and any *managed* attributes must be blocked from replication using fractional replication.

  Any changes that are made on one supplier will automatically trigger the plug-in to manage the values on the corresponding directory entries, so the data stay consistent across servers. However, the managed attributes must be maintained by the plug-in instance for the data to be consistent between the linked entries. This means that managed attribute values should be maintained solely by the plug-in processes, not the replication process, even in a multi-master replication environment.

For information on using fractional replication, see Section 11.1.7, "Replicating a Subset of Attributes with Fractional Replication".

5.3.2. Looking at the Linking Attributes Plug-in Syntax

The default Linked Attributes Plug-in entry is a container entry for each plug-in instance, similar to the password syntax plug-ins or the DNA Plug-in in the next section. Each entry beneath this container entry defines a different link-managed attribute pair.

To create a new linking attribute pair, then, create a new plug-in instance beneath the container entry. A basic linking attribute plug-in instance required defining two things:

- The attribute that is managed manually by administrators, in the ***linkType*** attribute
- The attribute that is created dynamically by the plug-in, in the ***managedType*** attribute
- Optionally, a scope that restricts the plug-in to a specific part of the directory tree, in the ***linkScope*** attribute

Example 5.4. Example Linked Attributes Plug-in Instance Entry

```
dn: cn=Manager Link,cn=Linked Attributes,cn=plugins,cn=config
objectClass: top
objectClass: extensibleObject
cn: Manager Link1
cn: Manager Link
linkType: directReport
managedType: manager
linkScope: ou=people,dc=example,dc=com
```

All of the attributes available for an instance of the Linked Attributes Plug-in instance are listed in Table 5.2, "Linked Attributes Plug-in Instance Attributes".

Table 5.2. Linked Attributes Plug-in Instance Attributes

Plug-in Attribute	Description
cn	Gives a unique name for the plug-in instance.
linkScope	Contains the DN of a suffix to which to restrict the function of the plug-in instance.
linkType	Gives the attribute which is maintained by an administrator. This attribute is manually maintained and is used as the reference for the plug-in. This attribute must have a DN value format. When the attribute is added, modified, or deleted, then its value contains the DN of the target entry for the plug-in to update.
managedType	Gives the attribute which is maintained by the plug-in. This attribute is created and updated on target entries. This attribute must have a DN value format. When the attribute is added to the entry, its value will point back as a cross-reference to the managed entry.

5.3.3. Configuring Attribute Links

NOTE

The Linked Attribute Plug-in instance can be created in the Directory Server Console, but only through the Advanced Property Editor for the directory entry, by manually adding all of the required attributes, the same as creating the entry manually through the command line.

1. If it is not already enabled, enable the Linked Attributes Plug-in, as described in Section 1.8.1, "Enabling Plug-ins in the Directory Server Console". For example:

   -

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=Linked Attributes,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Create the plug-in instance. Both the ***managedType*** and ***linkType*** attributes are required. The plug-in syntax is covered in Section 5.3.2, "Looking at the Linking Attributes Plug-in Syntax" . For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Manager Link,cn=Linked Attributes,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
cn: Manager Link
linkType: directReport
managedType: manager
```

3. Restart the server to apply the new plug-in instance.

```
service dirsrv restart
```

5.3.4. Cleaning up Attribute Links

The managed-linked attributes can get out of sync. For instance, a linked attribute could be imported or replicated over to a server, but the corresponding managed attribute was not because the link attribute was not properly configured. The managed-linked attribute pairs can be fixed by running a script (**fixup-linkedattrs.pl**) or by launching a fix-up task.

The fixup task removes any managed attributes (attributes managed by the plug-in) that do not have a corresponding link attribute (attributes managed by the administrator) on the referenced entry. Conversely, the task adds any missing managed attributes if the link attribute exists in an entry.

5.3.4.1. Regenerating Linked Attributes Using fixup-linkedattrs.pl

The **fixup-linkedattrs.pl** script launches a special task to regenerate all of the managed-link attribute pairs on directory entries. One or the other may be lost in certain situations. If the link attribute exists in an entry, the task traces the cross-referenced DN in the available attribute and creates the corresponding configured managed attribute on the referenced entry. If a managed attribute exists with no corresponding link attribute, then the managed attribute value is removed.

To repair all configured link attribute pairs for the entire scope of the plug-in, then simply run the command as the Directory Manager:

```
/usr/lib64/dirsrv/instance_name/fixup-linkedattrs.pl -D "cn=Directory Manager" -w password
```

It is also possible to limit the fixup task to a single link-managed attribute pair, using the **-l** option to specify the target plug-in instance DN:

```
/usr/lib64/dirsrv/instance_name/fixup-linkedattrs.pl -D "cn=Directory Manager" -w password -l "cn=Manager
Link,cn=Linked Attributes,cn=plugins,cn=config"
```

The **fixup-linkedattrs.pl** tool is described in more detail in the *Configuration and Command-Line Tool Reference* .

5.3.4.2. Regenerating Linked Attributes Using ldapmodify

Repairing linked attributes is one of the tasks which can be managed through a special task configuration entry. Task entries occur under the **cn=tasks** configuration entry in the **dse.ldif** file, so it is also possible to initiate a task by adding the entry using **ldapmodify**. When the task is complete, the entry is removed from the directory.

This task is the same one created automatically by the **fixup-linkedattrs.pl** script when it is run.

To initiate a linked attributes fixup task, add an entry under the **cn=fixup linked attributes,cn=tasks,cn=config** entry. The only required attribute is the ***cn*** for the specific task, though it also allows the ***ttl*** attribute to set a timeout period.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example,cn=fixup linked attributes,cn=tasks,cn=config
changetype: add
cn:example
ttl: 5
```

Once the task is completed, the entry is deleted from the **dse.ldif** configuration, so it is possible to reuse the same task entry continually.

The **cn=fixup linked attributes** task configuration is described in more detail in the *Configuration and Command-Line Tool Reference*.

5.4. ASSIGNING AND MANAGING UNIQUE NUMERIC ATTRIBUTE VALUES

Some entry attributes require having a unique number, such as ***uidNumber*** and ***gidNumber***. The Directory Server can automatically generate and supply unique numbers for specified attributes using the Distributed Numeric Assignment (DNA) Plug-in.

NOTE

Attribute uniqueness is not necessarily preserved with the DNA Plug-in. The plug-in only assigns non-overlapping ranges, but it does allow manually-assigned numbers for its managed attributes, and it does not verify or require that the manually-assigned numbers are unique.

The issue with assigning unique numbers is not with generating the numbers but in effectively avoiding replication conflicts. The DNA Plug-in assigns unique numbers across a *single* back end. For multi-master replication, when each master is running a local DNA Plug-in instance, there has to be a way to ensure that each instance is using a truly unique set of numbers. This is done by assigning different *ranges* of numbers to each server to assign.

5.4.1. About Dynamic Number Assignments

The DNA Plug-in for a server assigns a range of available numbers that that instance can issue. The range definition is very simple and is set by two attributes: the server's next available number (the low end of the range) and its maximum value (the top end of the range). The initial bottom range is set when the plug-in instance is configured. After that, the bottom value is updated by the plug-in. By breaking the available numbers into separate ranges on each replica, the servers can all continually assign numbers without overlapping with each other.

5.4.1.1. Filters, Searches, and Target Entries

The server performs a sorted search, internally, to see if the next specified range is already taken, requiring the managed attribute to have an equality index with the proper ordering matching rule (as described in Section 9.2, "Creating Standard Indexes").

The DNA Plug-in is applied, always, to a specific area of the directory tree (the *scope*) and to specific entry types within that subtree (the *filter*).

IMPORTANT

The DNA Plug-in only works on a single back end; it cannot manage number assignments for multiple databases. The DNA plug-in uses the sort control when checking whether a value has already been manually allocated outside of the DNA Plug-in. This validation, using the sort control, only works on a single back end.

5.4.1.2. Ranges and Assigning Numbers

There are several different ways that the Directory Server can handle generating attribute values:

- In the simplest case, a user entry is added to the directory with an object class which requires the unique-number attribute, but without the attribute present. Adding an entry with no value for the managed attribute triggers the DNA Plug-in to assign a value. This option only works if the DNA Plug-in has been configured to assign unique values to a single attribute.
- A similar and more manageable option is to use a *magic number*. This magic number is a template value for the managed attribute, something outside the server's range, a number or even a word, that the plug-in recognizes it needs to replace with a new assigned value. When an entry is added with the magic value and the entry is within the scope and filter of the configured DNA Plug-in, then using the magic number automatically triggers the plug-in to generate a new value. For example, this uses 0 as a magic number:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
uidNumber: 0
gidNumber: 0
....
```

The DNA Plug-in only generates new, unique values. If an entry is added or modified to use a specific value for an attribute controlled by the DNA Plug-in, the specified number is used; the DNA Plug-in will not overwrite it.

5.4.1.3. Multiple Attributes in the Same Range

The DNA Plug-in can assign unique numbers to a single attribute type or across multiple attribute types from a single range of unique numbers.

This provides several options for assigning unique numbers to attributes:

- A single number assigned to a single attribute type from a single range of unique numbers.
- The same unique number assigned to two attributes for a single entry.
- Two different attributes assigned two different numbers from the same range of unique numbers.

In many cases, it is sufficient to have a unique number assigned per attribute type. When assigning an ***employeeID*** to a new employee entry, it is important each employee entry is assigned a unique ***employeeID***.

However, there are cases where it may be useful to assign unique numbers from the same range of numbers to multiple attributes. For example, when assigning a ***uidNumber*** and a ***gidNumber*** to a **posixAccount** entry, the DNA Plug-in will assign the same number to both attributes. To do this, then pass both managed attributes to the modify operation, specifying the magic value.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
add: uidNumber
uidNumber: 0
-
add:gidNumber
gidNumber: 0
```

When multiple attributes are handled by the DNA Plug-in, the plug-in can assign a unique value to only one of those attributes if the object class only allows one of them. For example, the **posixGroup** object class does not allow a ***uidNumber*** attribute but it does allow ***gidNumber***. If the DNA Plug-in manages both ***uidNumber*** and ***gidNumber***, then when a **posixGroup** entry is created, a unique number for ***gidNumber*** is assigned from the same range as the ***uidNumber*** and ***gidNumber*** attributes. Using the same pool for all attributes manged by the plug-in keeps the assignment of unique numbers aligned and prevents situations where a ***uidNumber*** and a ***gidNumber*** on different entries are assigned from different ranges and result in the same *unique* number.

If multiple attributes are handled by the DNA Plug-in, then the same value will be assigned to all of the given managed attributes in an entry in a single modify operation. To assign *different* numbers from the same range, then you must perform separate modify operations. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
add: uidNumber
uidNumber: 0
^D

[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: uid=jsmith,ou=people,dc=example,dc=com
```

```
changetype: modify
add: employeeId
employeeId: magic
```

IMPORTANT

When the DNA Plug-in is configured to assign unique numbers to multiple attributes, it is necessary to specify the magic value for each attribute that requires the unique number. While this is not necessary when the DNA plug-in has been configured to provide unique numbers for a single attribute, it is necessary for multiple attributes. There may be instances where an entry does not allow each type of attribute defined for the range, or, more important, an entry allow all of the attributes types defined, but only a subset of the attributes require the unique value.

Example 5.5. DNA and Unique Bank Account Numbers

Example Bank wants to use the same unique number for a customer's ***primaryAccount*** and ***customerID*** attributes. The Example Bank administrator configured the DNA Plug-in to assign unique values for both attributes from the same range.

The bank also wants to assign numbers for secondary accounts from the same range as the customer ID and primary account numbers, but these numbers cannot be the same as the primary account numbers. The Example Bank administrator configures the DNA Plug-in to also manage the ***secondaryAccount*** attribute, but will only add the ***secondaryAccount*** attribute to an entry *after* the entry is created and the ***primaryAccount*** and ***customerID*** attributes are assigned. This ensures that ***primaryAccount*** and ***customerID*** share the same unique number, and any ***secondaryAccount*** numbers are entirely unique but still from the same range of numbers.

5.4.2. Looking at the DNA Plug-in Syntax

The DNA Plug-in itself is a container entry, similar to the Password Storage Schemes Plug-in. Each DNA entry underneath the DNA Plug-in entry defines a new managed range for the DNA Plug-in.

To set new managed ranges for the DNA Plug-in, create entries beneath the container entry.

The most basic configuration is to set up distributed numeric assignments on a single server, meaning the ranges will not be shared or transferred between servers. A basic DNA configuration entry defines four things:

- The attribute that value is being managed, is set in the ***dnaType*** attribute
- The entry DN to use as the base to search for entries, set in the ***dnaScope*** attribute
- The search filter to use to identify entries to manage, set in the ***dnaFilter*** attribute
- The next available value to assign, set in the ***dnaNextValue*** attribute (after the entry is created, this is handled by the plug-in)

All of the attributes available for a DNA Plug-in instance are listed in Table 5.3, “DNA Entry Attributes” .

To configure distributed numeric assignment on a single server for a single attribute type:

```
dn: cn=Account UIDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
objectClass: top
objectClass: dnaPluginConfig
cn: Account UIDs
dnatype: uidNumber
dnafilter: (objectclass=posixAccount)
dnascope: ou=people,dc=example,dc=com
dnaNextValue: 1
```

If multiple suppliers are configured for distributed numeric assignments, then the entry must contain the required information to transfer ranges:

- The maximum number that the server can assign; this sets the upward bound for the range, which is logically required when multiple servers are assigning numbers. This is set in the ***dnaMaxValue*** attribute.
- The threshold where the range is low enough to trigger a range transfer, set in the ***dnaThreshold*** attribute. If this is not set, the default value is **1**.
- A timeout period so that the server does not hang waiting for a transfer, set in the ***dnaRangeRequestTimeout*** attribute. If this is not set, the default value is **10**, meaning 10 seconds.

- A configuration entry DN which is shared among all supplier servers, which stores the range information for each supplier, set in the ***dnaSharedCfgDN*** attribute.

The specific number range which could be assigned by the server is defined in the ***dnaNextRange*** attribute. This shows the next available range for transfer and is managed automatically by the plug-in, as ranges are assigned or used by the server. This range is just "on deck." It has not yet been assigned to another server and is still available for its local Directory Server to use.

```
dn: cn=Account UIDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
objectClass: top
objectClass: dnaPluginConfig
cn: Account UIDs
dnatype: uidNumber
dnafilter: (objectclass=posixAccount)
dnascope: ou=People,dc=example,dc=com
dnanextvalue: 1
dnaMaxValue: 1300
dnasharedcfgdn: cn=Account UIDs,ou=Ranges,dc=example,dc=com
dnathreshold: 100
dnaRangeRequestTimeout: 60
dnaNextRange: 1301-2301
```

The ***dnaNextRange*** attribute should be set explicitly only if a separate, specific range has to be assigned to other servers. Any range set in the ***dnaNextRange*** attribute must be unique from the available range for the other servers to avoid duplication. If there is no request from the other servers and the server where ***dnaNextRange*** is set explicitly has reached its set ***dnaMaxValue***, the next set of values (part of the ***dnaNextRange***) is allocated from this deck.

The ***dnaNextRange*** allocation is also limited by the ***dnaThreshold*** attribute that is set in the DNA configuration. Any range allocated to another server for ***dnaNextRange*** cannot violate the threshold for the server, even if the range is available on the deck of ***dnaNextRange***.

NOTE

If the ***dnaNextRange*** attribute is handled internally if it is not set explicitly. When it is handled automatically, the ***dnaMaxValue*** attribute serves as upper limit for the next range.

Each supplier keeps a track of its current range in a separate configuration entry which contains information about its range and its connection settings. This entry is a child of the location in ***dnasharedcfgdn***. The configuration entry is replicated to all of the other suppliers, so each supplier can check that configuration to find a server to contact for a new range. For example:

```
dn: dnaHostname=ldap1.example.com+dnaPortNum=389,cn=Account UIDs,ou=Ranges,dc=example,dc=com
objectClass: dnaSharedConfig
objectClass: top
dnahostname: ldap1.example.com
dnaPortNum: 389
dnaSecurePortNum: 636
dnaRemainingValues: 1000
```

Table 5.3. DNA Entry Attributes

Plug-in Attribute	Description
dnaPluginConfig (object class)	The object class for instances of the DNA Plug-in.
cn	Gives a unique name for the plug-in instance.
dnaType	Contains the name of the attributes for which unique numbers are assigned. If a prefix will be prepended to the generated value, then be sure to use an attribute which allows the syntax of the combined attribute value, such as a custom attribute which allows alphanumeric strings. Otherwise, syntax validation will enforce the defined syntax for the value, such as integer for ***uidNumber*** and ***gidNumber***, and the DNA operations will fail with syntax violations.

Plug-in Attribute	Description
dnaScope	Sets the base DN to use to search for entries to which to apply the managed ranges.
dnaFilter	Gives an LDAP filter to use to specify the kinds of entries for the plug-in to manage.
dnaNextValue	Gives the next available number to assign. This is initially set manually when the entry is created; afterward, it is managed by the plug-in.
dnaMaxValue	Optionally, the upper limit of the range that the server can assign. Defining the range is required when there are multiple servers assigning numbers to entries. The default value is **-1**, which is the same as the highest 64-bit integer.
dnaInterval	Optionally, sets an interval to use to increment through numbers in a range. Essentially, this skips numbers at a predefined rate. If the interval is 3 and the first number in the range is 1, then the next number used in the ragen is 4, then 7, then 10, incrementing by three for every new number assignment.
dnaThreshold	Sets a limit on the amount of remaining available numbers before the server requests a new range.
dnaSharedCfgDN	Specifies the DN of a container entry that each supplier server shares. The plug-in automatically creates an entry for the individual instances underneath this entry which contains their available ranges. The plug-in can use this information to request and transfer ranges as servers consume their available range.
dnaNextRange	Shows the next range of numbers which are available to be transferred. This attribute can be set automatically by the plug-in according to the threshold and shared configuration information; this can also be set manually for an administrator to specifically assign an additional range of values to a server. This attribute is always limited by the **dnaThreshold** settings.
dnaRangeRequestTimeout	Sets a timeout period for a range request so that a server does not hang indefinitely waiting for a transfer.
dnaMagicRegen	Sets a word or number (outside of the assigned range) which automatically triggers the plug-in to assign a number to an attribute. This is a useful default to use for importing entries.
dnaPrefix	Sets a string to insert in front of whatever number is assigned. For example, if the prefix is **user** and the assigned number for the attribute is **1001**, then the final assigned value is **user1001**. ***dnaPrefix*** can hold any kind of string. However, some possible values for ***dnaType*** (such as ***uidNumber*** and ***gidNumber***) require integer syntax for attribute values. To use a prefix string, consider using a custom attribute for ***dnaType*** which allows the syntax of the prefix plus the generated number assignment.
dnaSharedConfig (object class)	The object class for shared configuration entries with host information, for servers in multi-master replication.

Plug-in Attribute	Description
dnaHostname	Identifies the host name of a server in a shared range, as part of the DNA range configuration for that specific host in multi-master replication.
dnaPortNum	Gives the standard port number to use to connect to the host identified in ***dnaHostname***.
dnaSecurePortNum	Gives the secure (SSL) port number to use to connect to the host identified in ***dnaHostname***.
dnaRemainingValues	Contains the number of values that are remaining and available to a server to assign to entries.

5.4.3. Configuring Unique Number Assignments

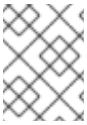
The unique number distribution is configured by creating different instances of the DNA Plug-in. These DNA Plug-in instances can only be created through the command line, but they can be edited through the Directory Server Console.

5.4.3.1. Configuring Unique Number Assignments

NOTE

Any attribute which has a unique number assigned to it must have an equality index set for it. The server must perform a sorted search, internally, to see if the ***dnaNextvalue*** is already taken, which requires an equality index on an integer attribute, with the proper ordering matching rule.

Creating indexes is described in Section 9.2, “Creating Standard Indexes”.



NOTE

Set up the DNA Plug-in on every supplier server, and be careful not to overlap the number range values.

1. Create the shared container entry in the replicated subtree. For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: ou=Ranges,dc=example,dc=coma
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: organizationalUnit
ou: Ranges


dn: cn=Account UIDs,ou=Ranges,dc=example,dc=coma
changetype: add
objectclass: top
objectclass: extensibleObject
cn: Account UIDs
```

2. Enable the DNA Plug-in. By default, the plug-in entry (which is the container entry) is disabled.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

3. Create the new DNA Plug-in instance beneath the container entry. Table 5.3, “DNA Entry Attributes” lists the possible plug-in attributes.

NOTE

The plug-in attribute which sets which entry attributes have unique number assignments, ***dnaType***, is multi-valued. If multiple attributes are set in the same plug-in instance, then their number assignments are taken from the same range. To use different ranges, configure different plug-in instances.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Account UIDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: dnaPluginConfig
cn: Account UIDs
dnatype: uidNumber
dnafilter: (objectclass=posixAccount)
dnascope: ou=People,dc=example,dc=com
dnanextvalue: 1
dnaMaxValue: 1300
dnasharedcfgdn: cn=Account UIDs,ou=Ranges,dc=example,dc=com
dnathreshold: 100
dnaRangeRequestTimeout: 60
dnaMagicRegen: magic
```

4. For servers in multi-master replication, create a configuration entry for the host, which specifies its connection information and range.

   The DN of the entry is a combination of the host name and the port number (**dnaHostname+dnaPortNum**).

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: dnaHostname=ldap1.example.com+dnaPortNum=389,cn=Account
UIDs,ou=Ranges,dc=example,dc=com
objectClass: dnaSharedConfig
objectClass: top
dnahostname: ldap1.example.com
dnaPortNum: 389
dnaSecurePortNum: 636
dnaRemainingValues: 1000
```

5. Restart the server to load the new plug-in instance.

```
service dirsrv restart instance_name
```

5.4.3.2. Editing the DNA Plug-in in the Console

NOTE

Any attribute which has a unique number assigned to it must have an equality index set for it. The server must perform a sorted search, internally, to see if the ***dnaNextvalue*** is already taken, which requires an equality index on an integer attribute, with the proper ordering matching rule.

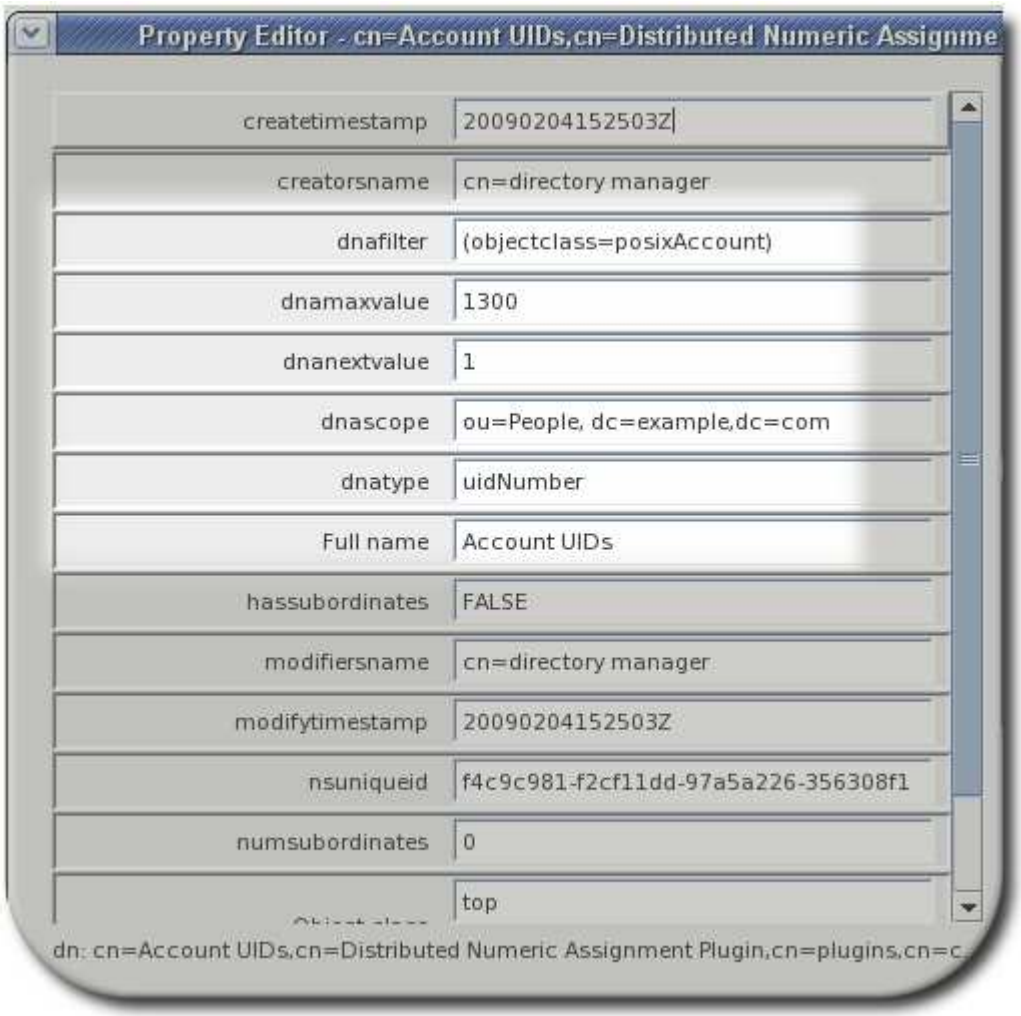
Creating indexes is described in Section 9.2, “Creating Standard Indexes”.

The Directory Server Console can be used to edit the DNA Plug-in instances.

1. Click the **Directory** tab.
2. Open the **config** folder, and then expand the **plugins** folder.
3. Click the **Distributed Numeric Assignment** plug-in folder. All of the DNA Plug-in instances are listed in the main window.

4. Highlight the DNA instance entry, and right-click on the **Advanced** link to open the property editor.

5. Edit the DNA-related attributes.



5.4.4. Distributed Number Assignment Plug-in Performance Notes

There can be thread locking issues as DNA configuration is changed dynamically, so that new operations which access the DNS configuration (such as a DNA task or additional changes to the DNA configuration) will access the old configuration because the thread with the new configuration has not yet been released. This can cause operations to use old configuration or simply cause operations to hang.

To avoid this, preserve an interval between dynamic DNA configuration changes of 35 seconds. This means have a sleep or delay between both DNA configuration changes and any directory entry changes which would trigger a DNA plug-in operation.

CHAPTER 6. ORGANIZING AND GROUPING ENTRIES

Entries contained within the directory can be grouped in different ways to simplify the management of user accounts. Red Hat Directory Server supports a variety of methods for grouping entries and sharing attributes between entries. To take full advantage of the features offered by roles and class of service, determine the directory topology when planning the directory deployment.

6.1. USING GROUPS

Groups are a mechanism for associating entries for ease of administration. Groups do not have the flexibility or utility of roles. However, there are certain clients and applications where groups are useful. Groups also offer compatibility with older LDAP clients and directory services.

NOTE

Managing groups is significantly easier by using the ***memberOf*** attribute to identify in user entries to what groups a user belongs. The ***memberOf*** attribute is maintained by the Directory Server and updated automatically on entries as group membership changes. See Section 6.1.4, "Listing Group Membership in User Entries" for information on using the ***memberOf*** attribute.

6.1.1. Creating Static Groups in the Console

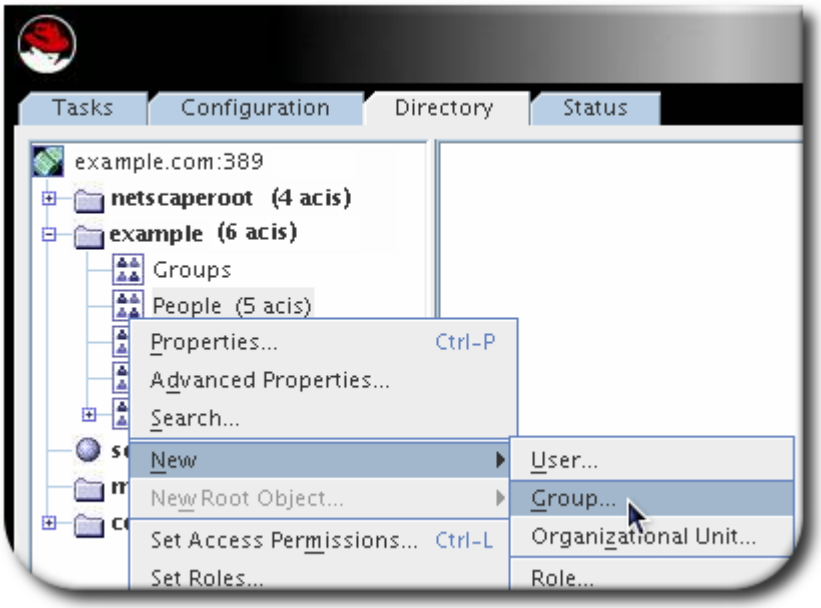
Static groups organize entries by specifying the same group value in the DN attribute of any number of users.

NOTE

If a user has an entry on a remote Directory Server (for example, in a chained database), different from the Directory Server which has the entry that defines the static group, then use the Referential Integrity plug-in to ensure that deleted user entries are automatically deleted from the static group.

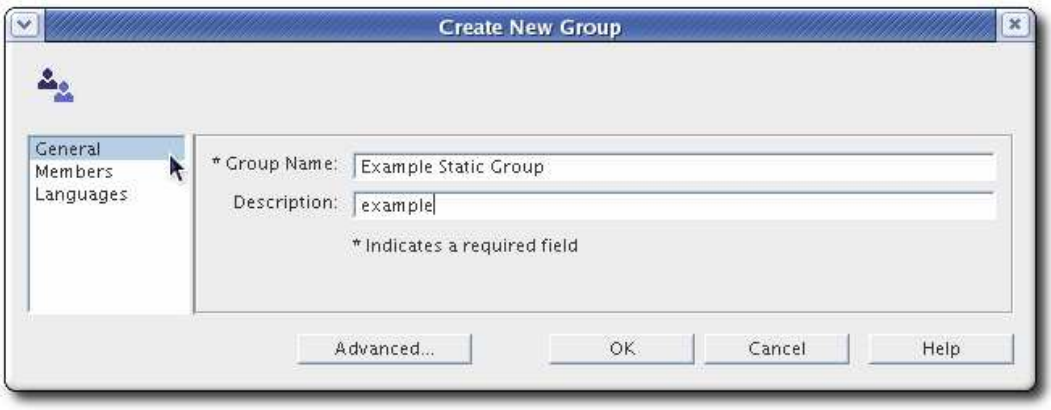
There are some performance and access control considerations with the Referential Integrity plug-in. For more information about using referential integrity with chaining, see Section 2.3.2, "Configuring the Chaining Policy".

1. In the Directory Server Console, select the **Directory** tab.
2. In the left pane, right-click the entry under which to add a new group, and select **New > Group**.



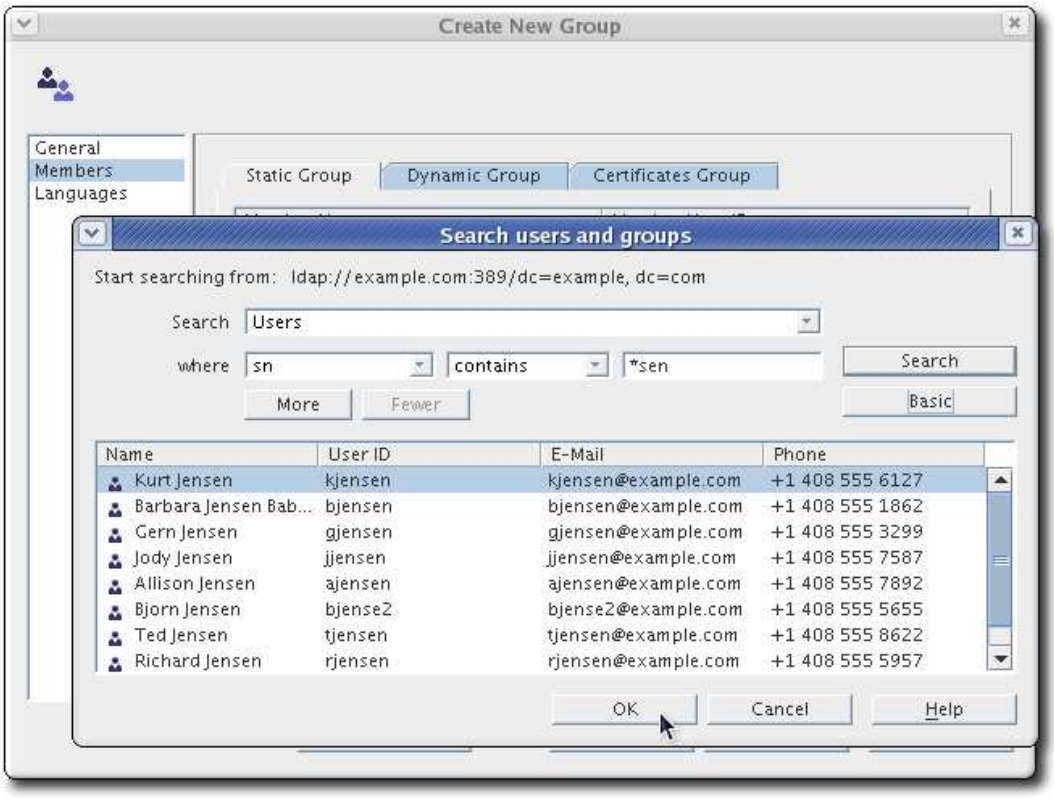
   Alternatively, go to the **Object** menu, and select **New > Group**.

3. Click **General** in the left pane. Type a name for the new group in the **Group Name** field (the name is required), and enter a description of the new group in the **Description** field.

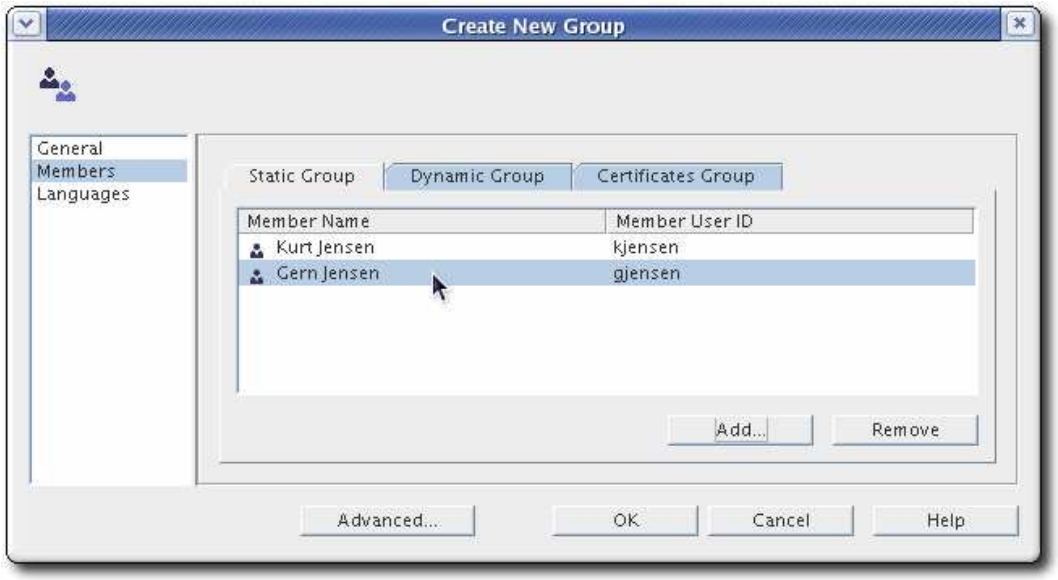


4. Click **Members** in the left pane. In the right pane, select the **Static Group** tab. Click **Add** to add new members to the group.

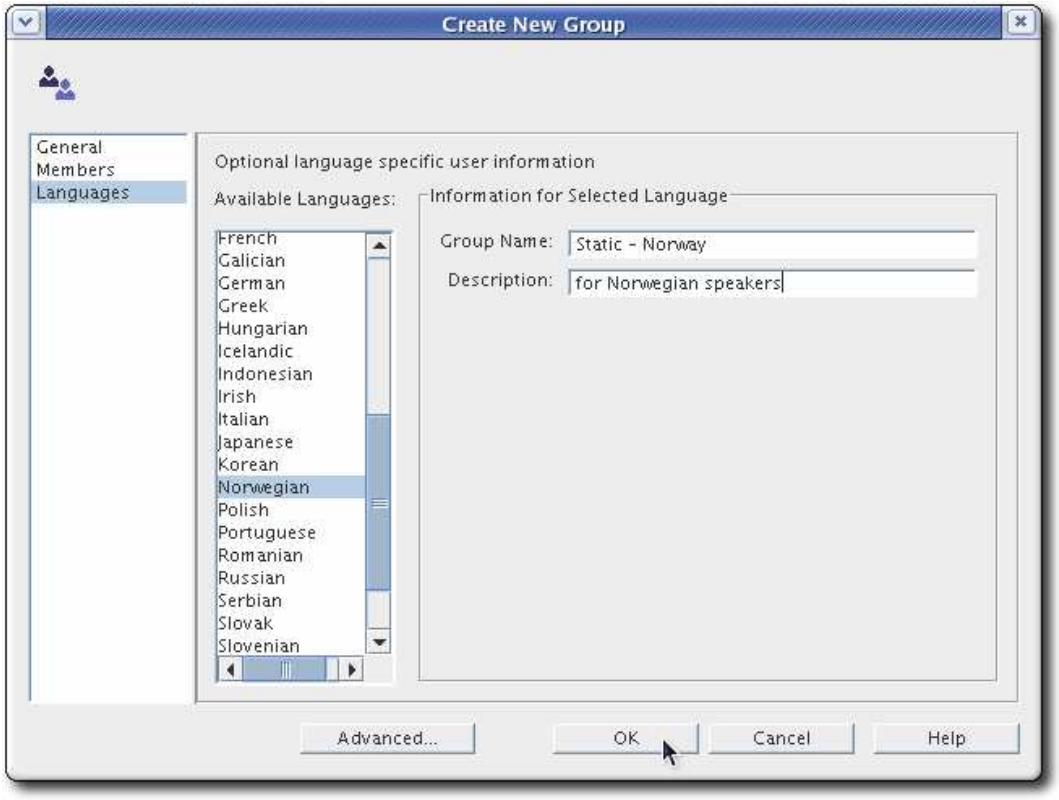
5. In the **Search** drop-down list, select what sort of entries to search for (users, groups, or both) then click **Search**.



6. Select the members from the returned entries, and click **OK**.



7. Click **Languages** in the left pane to add language-specific information for the group.



8. Click **OK** to create the new group. It appears in the right pane.

To edit a static group, double-click the group entry, and make the changes in the editor window. To view the changes, go to the **View** menu, and select **Refresh**.

NOTE

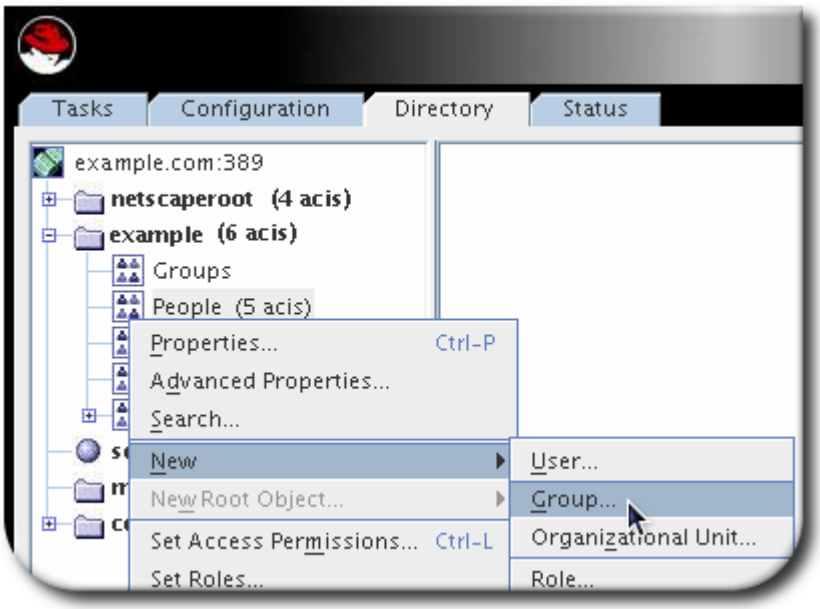
The Console for managing static groups may not display all possible selections during a search operation if there is no VLV index for users' search. This problem occurs only when the number of users is 1000 or more and there is no VLV index for search. To work around the problem, create a VLV index for the users suffix with the filter **(objectclass=person)** and scope **sub-tree**.

6.1.2. Creating Dynamic Groups in the Console

Dynamic groups filter users based on their DN and include them in a single group.

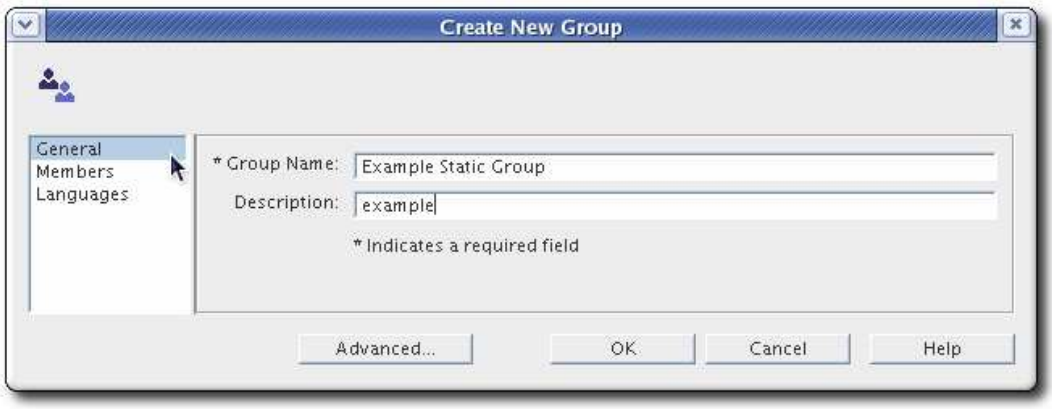
1. In the Directory Server Console, select the **Directory** tab.

2. In the left pane, right-click the entry under which to add a new group, and select **New > Group**.



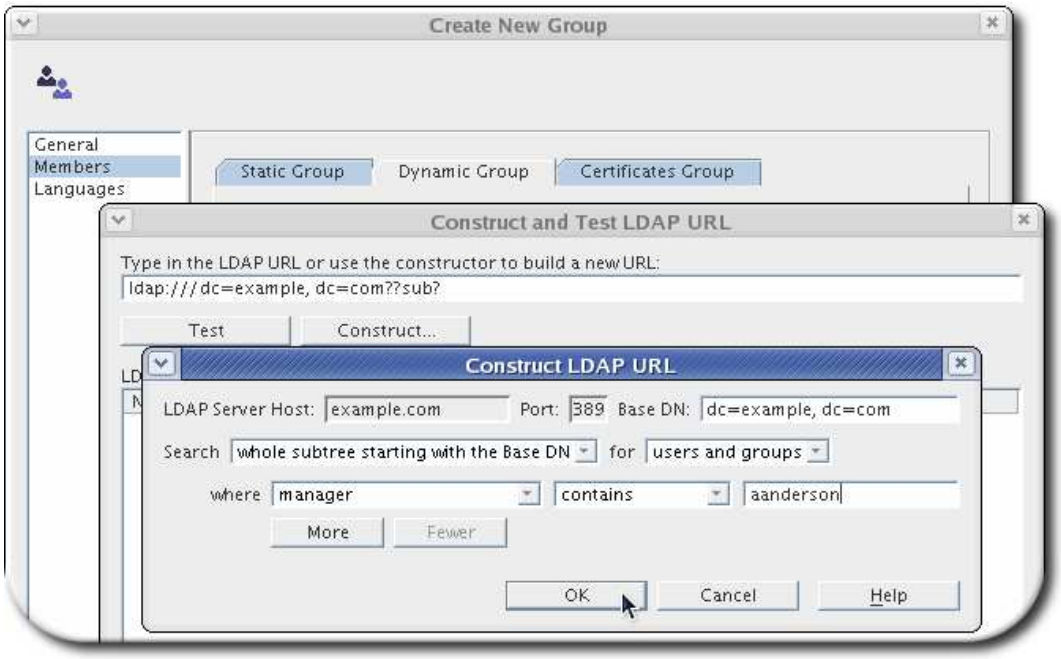
Alternatively, go to the **Object** menu, and select **New > Group**.

3. Click **General** in the left pane. Type a name for the new group in the **Group Name** field (the name is required), and enter a description of the new group in the **Description** field.

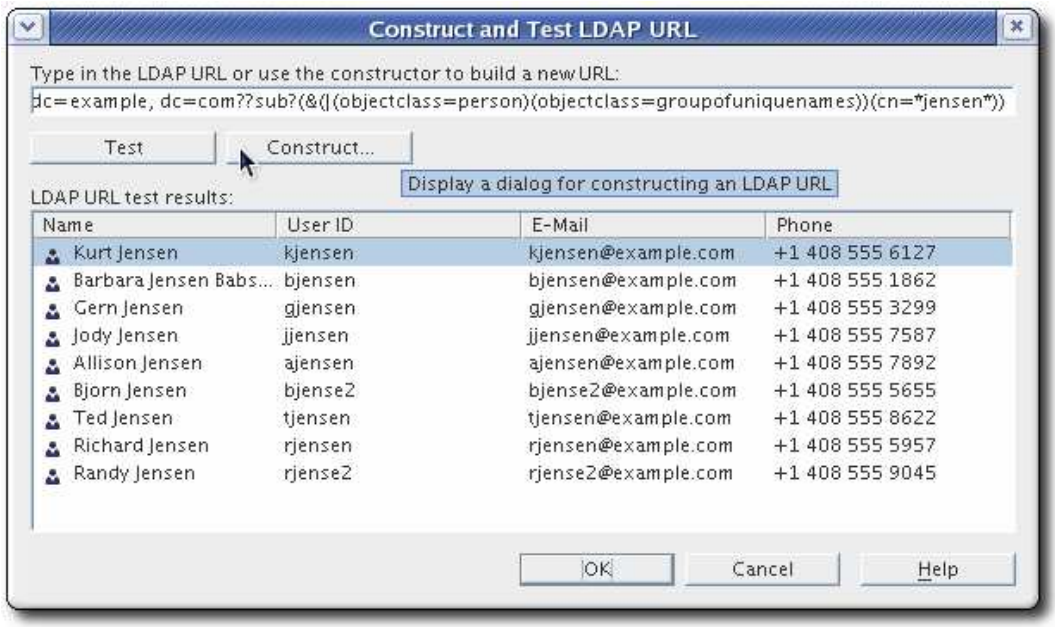


4. Click **Members** in the left pane. In the right pane, select the **Dynamic Group** tab. Click **Add** to create a LDAP URL for querying the database.

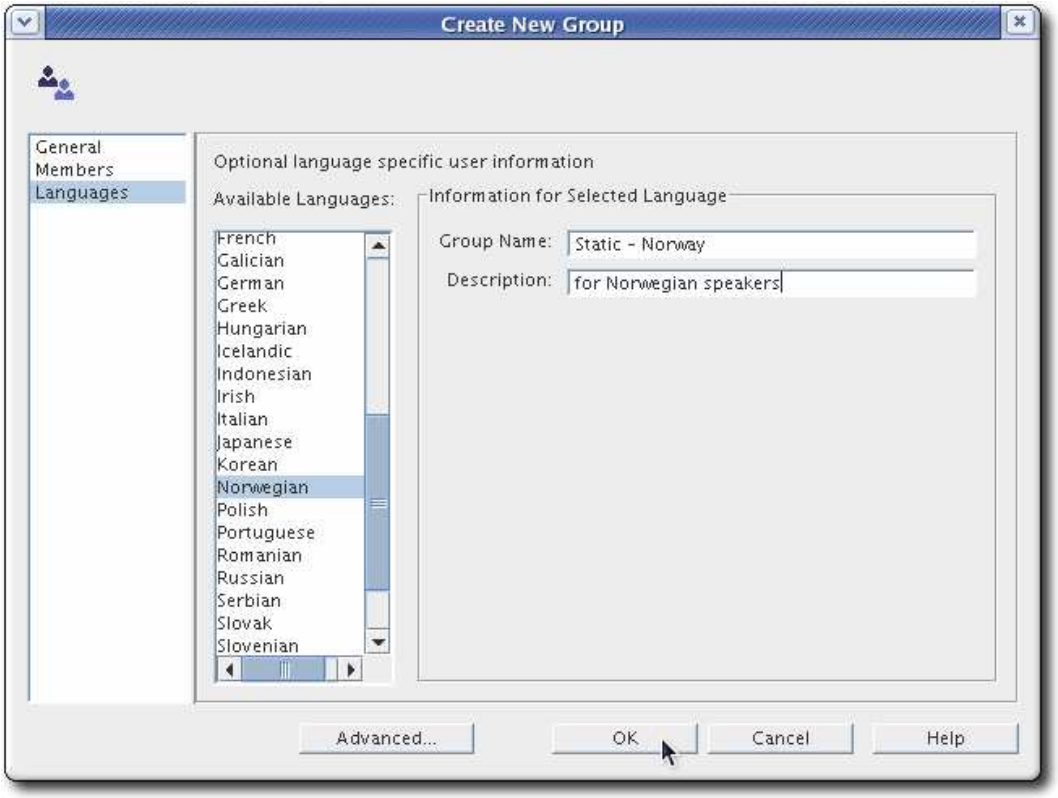
5. Enter an LDAP URL in the text field or select **Construct** to be guided through the construction of an LDAP URL.



The results show the current entries (group members) which correspond to the filter.



6. Click **Languages** in the left pane to add language-specific information for the group.



7. Click **OK**. The new group appears in the right pane.

To edit a dynamic group, double-click the group entry to open the editor window, and make whatever changes to the dynamic group. To view the changes to the group, go to the **View** menu, and select **Refresh**.

NOTE

The Console for managing dynamic groups may not display all possible selections during a search operation if there is no VLV index for users' search. This problem can occur when the number of users is 1000 or more and there is no VLV index for search. To work around the problem, create a VLV index for the users suffix with the filter **(objectclass=person)** and scope **sub-tree**.

6.1.3. Creating Groups in the Command Line

Creating both static and dynamic groups from the command line is a similar process. A group entry contains the group name, the type of group, and a members attribute.

There are several different options for the type of group; these are described in more detail in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*. The *type of group* in this case refers to the type of defining member attribute it has:

- **groupOfNames** is a simple group, that allows any entry to be added. The attribute used to determine members for this is ***member***.
- **groupOfUniqueNames**, like **groupOfNames**, simply lists user DNs as members, but the members must be unique. This prevents users being added more than once as a group member, which is one way of preventing self-referential group memberships. The attribute used to determine members for this is ***uniqueMember***.
- **groupOfURLs** uses a list of LDAP URLs to filter and generate its membership list. This object class is required for any dynamic group and can be used in conjunction with **groupOfNames** and **groupOfUniqueNames**.
- **groupOfCertificates** is similar to **groupOfURLs** in that it uses an LDAP filter to search for and identify certificates (or, really, certificate names) to identify group members. This is useful for group-based access control, since the group can be given special access permissions. The attribute used to determine members for this is ***memberCertificate***.

Table 6.1, "Dynamic and Static Group Schema" lists the default attributes for groups as they are created from the command line.

Table 6.1. Dynamic and Static Group Schema

Type of Group	Group Object Classes	Member Attributes
Static	groupOfUniqueNames	uniqueMember
Dynamic	groupOfUniqueNames groupOfURLs	memberURL

A static group entry lists the specific members of the group. For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=static group,ou=Groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfUniqueNames
cn: static group
description: Example static group.
uniqueMember: uid=mwhite,ou=People,dc=example,dc=com
uniqueMember: uid=awhite,ou=People,dc=example,dc=com
```

A dynamic group uses at least one LDAP URL to identify entries belonging to the group and can specify multiple LDAP URLs or, if used with another group object class like **groupOfUniqueNames**, can explicitly list some group members along with the dynamic LDAP URL.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=dynamic group,ou=Groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfUniqueNames
objectClass: groupOfURLs
cn: dynamic group
description: Example dynamic group.
memberURL: ldap:///dc=example,dc=com??sub?(&(objectclass=person)(cn=*sen*))
```

6.1.4. Listing Group Membership in User Entries

The entries which belong to a group are defined, in some way, in the group entry itself. This makes it very easy to look at a group and see its members and to manage group membership centrally. However, there is no good way to find out what groups a single user belongs to. There is nothing in a user entry which indicates its memberships, as there are with roles.

The MemberOf Plug-in correlates group membership lists to the corresponding user entries.

The MemberOf Plug-in analyzes the member attribute in a group entry and automatically writes a corresponding ***memberOf*** attribute in the member's entry. (By default, this checks the ***member*** attribute, but mutiple attribute instances can be used to support multiple different group types.)

As membership changes, the plug-in updates the ***memberOf*** attributes on the user entries. The MemberOf Plug-in provides a way to view the groups to which a user belongs simply by looking at the entry, including nested group membership. It can be very difficult to backtrack memberships through nested groups, but the MemberOf Plug-in shows memberships for all groups, direct and indirect.

The MemberOf Plug-in manages member attributes for static groups, not dynamic groups or circular groups.

6.1.4.1. Directory Topology Considerations with the MemberOf Plug-in

memberOf and Multi-Master Replication

The ***memberOf*** attributes for user entries *should not be replicated in multi-master environments*. Make sure that the ***memberOf*** attribute is excluded from replication in the replication agreement. (Fractional replication is described in Section 11.1.7, “Replicating a Subset of Attributes with Fractional Replication” .)

Each server must maintain its own MemberOf Plug-in independently. To make sure that the ***memberOf*** attributes for entries are the same across servers, simply configure the MemberOf Plug-in the same on all servers.

With single-master replication, it is perfectly safe to replicate ***memberOf*** attributes. Configure the MemberOf Plug-in for the supplier, then replicate the ***memberOf*** attributes to the consumers.

memberOf and Distributed Databases

It is possible, as outlined in Section 2.2.1, "Creating Databases", to distribute suffixes and directory data across different databases.

By default, the MemberOf Plug-in only looks for potential members for users who are in the same database as the group. If users are stored in a different database than the group, then the user entries will not be updated with ***memberOf*** attributes because the plug-in cannot ascertain the relationship between them.

The plug-in can be configured to search across all databases by enabling the ***memberOfAllBackends*** in the plug-in configuration.

6.1.4.2. Required Object Classes by the memberOf Plug-In

To enable the **memberOf** plug-in to add the ***memberOf*** attribute, the user entry must contain the **inetUser** or **inetAdmin** object class to support this attribute. If you configure the **memberOf** plug-in to use a different attribute, make sure that the user entry contains an object class that supports this attribute.

To configure nested groups, the group must use the **extensibleObject** object class.

NOTE

If directory entries do not contain an object class that supports the required attributes, operations fail with the following error:

```
LDAP: error code 65 - Object Class Violation
```

6.1.4.3. The MemberOf Plug-in Syntax

The MemberOf Plug-in instance defines two attributes, one for the group member attribute to poll (***memberOfGroupAttr***) and the other for the attribute to create and manage in the member's user entry (***memberOfAttr***).

The ***memberOfGroupAttr*** attribute is multi-valued. Because different types of groups use different member attributes, using multiple ***memberOfGroupAttr*** attributes allows the plug-in to manage multiple types of groups.

The plug-in instance also gives the plug-in path and function to identify the MemberOf Plug-in and contains a state setting to enable the plug-in, both of which are required for all plug-ins. The default MemberOf Plug-in is shown in Example 6.1, "Default MemberOf Plug-in Entry" and the different parameters are described in Table 6.2, "MemberOf Syntax".

Example 6.1. Default MemberOf Plug-in Entry

```
dn: cn=MemberOf Plugin,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: MemberOf Plugin
nsslapd-pluginPath: libmemberof-plugin
nsslapd-pluginInitfunc: memberof_postop_init
nsslapd-pluginType: postoperation
nsslapd-pluginEnabled: on
nsslapd-plugin-depends-on-type: database
memberOfGroupAttr: member
memberOfGroupAttr: uniqueMember
memberOfAttr: memberOf
memberOfAllBackends: on
nsslapd-pluginId: memberOf
nsslapd-pluginVersion: 9.0.4
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: memberOf plugin
```

Table 6.2. MemberOf Syntax

Plug-in Attribute	Description

Plug-in Attribute	Description
cn	Gives a unique name for the plug-in instance.
memberOfGroupAttr	Gives the attribute in the group entry to poll to identify member DNs. By default, this is the ***member*** attribute, but it can be any attribute used to identify group members, such as ***uniqueMember***. This is a multi-valued attribute, so if multiple types of groups will be used with the MemberOf Plug-in, multiple member type attributes can be set.
memberOfAttr	Gives the attribute for the plug-in to create and manage on the user entry. By default, this is the ***memberOf*** attribute.
memberOfAllBackends	Sets whether the plug-in should evaluate user entries only within the local suffix (off) or whether it should evaluate all configured databases (on).
memberOfEntryScope	Sets on which suffixes the plug-in works on. If not set, the plug-in works on all suffixes.
memberOfEntryScopeExcludeSubtree	Sets what suffixes the plug-in excludes.
nsslapd-pluginEnabled	Sets whether the plug-in instance is enabled (active) or disabled. The default MemberOf Plug-in instance is disabled by default.
nsslapd-pluginPath	Gives the name of the specific plug-in to load. This must be **libmemberof-plugin**.
nsslapd-pluginInitfunc	Gives the name of the function to call to initialize the plug-in. This must be **memberof_postop_init**.

NOTE

To maintain backwards compatibility with older versions of Directory Server, which only allowed a single member attribute (by default, ***member***), it may be necessary to include the ***member*** group attribute or whatever previous member attribute was used, in addition any new member attributes used in the plug-in configuration.

```
memberOfGroupAttr: member
memberOfGroupAttr: uniqueMember
```

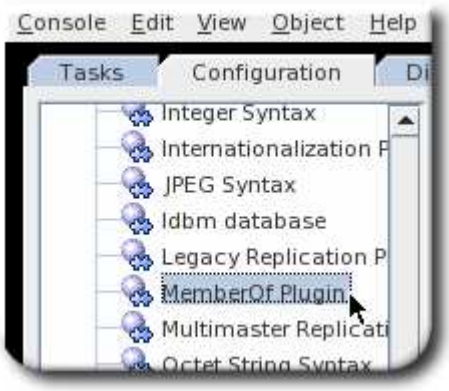
See Example 6.1, “Default MemberOf Plug-in Entry” .

6.1.4.4. Configuring an Instance of the MemberOf Plug-in

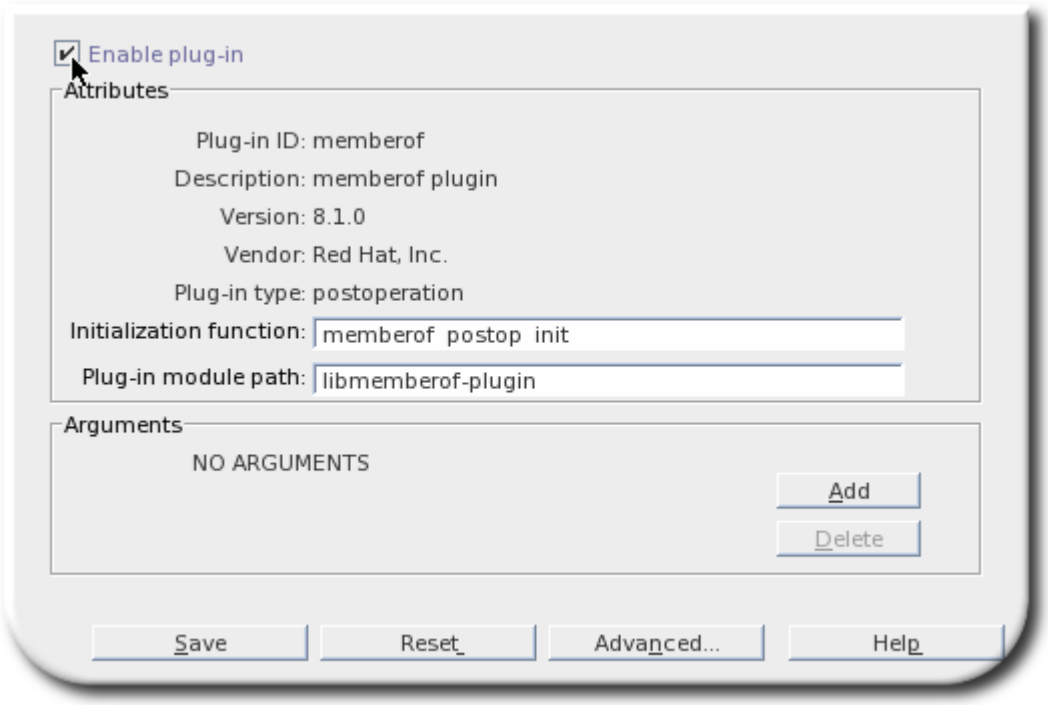
The attributes defined in the MemberOf Plug-in can be changed, depending on the types of groups used in the directory.

6.1.4.4.1. Editing the MemberOf Plug-in from the Console

1. Select the **Configuration** tab, and expand to the **Plugins** folder.
2. Scroll to the **Memberof Plugin** entry.



3. Make sure that the plug-in is enabled. This is disabled by default.



4. Click the **Advanced** button to open the **Advanced Properties Editor**.

5. The ***memberOfGroupAttr*** attribute sets the attribute in the group entry which the server uses to identify member entries; this attribute can be used multiple times for different group/member types. The ***memberOfAttr*** attribute sets the attribute which the plug-in creates and manages on user entries.

6. Save the changes.

7. Restart the server to update the plug-in. For example, open the **Tasks** tab and click the **Restart server** task.

6.1.4.4.2. Editing the MemberOf Plug-in from the Command Line

1. Enable the MemberOf Plug-in.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=MemberOf Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
-
```

2. Set the attribute to use for the group member entry attribute. The default attribute is ***member***, which can be changed using the **replace** command, or, since the ***memberOfGroupAttr*** attribute is multi-valued, additional member types can be added to the definition. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=MemberOf Plugin,cn=plugins,cn=config
changetype: modify
add: memberOfGroupAttr
memberOfGroupAttr: uniqueMember

add: memberOfGroupAttr
memberOfGroupAttr: customMember-
```

3. Set the attribute to set on the user entries to show group membership. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=MemberOf Plugin,cn=plugins,cn=config
changetype: modify
replace: memberOfAttr
memberOfAttr: memberOf
-
```

4. *Optional.* If the deployment uses distributed databases, then enable the ***memberOfAllBackends*** attribute to search through all databases, not just the local one, for user entries.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=MemberOf Plugin,cn=plugins,cn=config
changetype: modify
replace: memberOfAllBackends
memberOfAllBackends: on
-
```

5. Restart the Directory Server to load the modified new plug-in instance.

```
service dirsrv restart instance_name
```

6.1.4.5. Setting the Scope of the MemberOf Plug-in

If you configured several back ends or multiple-nested suffixes, you can use the ***memberOfEntryScope*** and ***memberOfEntryScopeExcludeSubtree*** parameters to set what suffixes the **MemberOf** plug-in works on.

If you add a user to a group, the **MemberOf** plug-in only adds the ***memberOf*** attribute to the group if both the user and the group are in the plug-in's scope. For example, to configure the **MemberOf** plug-in to work on all entries in ***dc=example,dc=com***, but to exclude entries in ***ou=private,dc=example,dc=com***, set:

```
memberOfEntryScope: dc=example,dc=com
memberOfEntryScopeExcludeSubtree: ou=private,dc=example,dc=com
```

If you moved a user entry out of the scope set in the ***memberOfEntryScope*** parameter:

- The membership attribute, such as ***member***, is updated in the group entry to remove the user DN value.
- The ***memberOf*** attribute is updated in the user entry to remove the group DN value.

NOTE

The value set in the ***memberOfEntryScopeExcludeSubtree*** parameter has a higher priority than values set in ***memberOfEntryScope***. If the scopes set in both parameters overlap, the **MemberOf** plug-in only works on the non-overlapping directory entries.

6.1.4.6. Synchronizing memberOf Values

The MemberOf Plug-in automatically manages the ***memberOf*** attribute on group member entries, based on the configuration in the group entry itself. However, the ***memberOf*** attribute can be edited on a user entry directly (which is improper) or new entries can be imported or replicated over to the server that have a ***memberOf*** attribute already set. These situations create inconsistencies between the ***memberOf*** configuration managed by the server plug-in and the actual memberships defined for an entry.

Directory Server has a ***memberOf*** repair task which manually runs the plug-in to make sure the appropriate ***memberOf*** attributes are set on entries. There are three ways to trigger this task:

- In the Directory Server Console
- Using the **fixup-memberof.pl** script
- Running a **cn=memberof task,cn=tasks,cn=config** tasks entry

NOTE

The memberOf regeneration tasks are run locally, even if the entries themselves are replicated. This means that the ***memberOf*** attributes for the entries on other servers are not updated until the updated entry is replicated.

6.1.4.6.1. Initializing and Regenerating memberOf Attributes Using fixup-memberof.pl

The **fixup-memberof.pl** script launches a special task to regenerate all of the ***memberOf*** attributes on user entries based on the defined member attributes in the group entries. This is a clean-up task which synchronizes the membership defined in group entries and the corresponding user entries and overwrites any accidental or improper edits on the user entries.

1. Open the tool directory for the Directory Server instance, **/usr/lib/dirsrv/slapd-*instance_name*/**.
2. Run the script, binding as the Directory Manager.

```
./fixup-memberof.pl -D "cn=Directory Manager" -w password
```

The **fixup-memberof.pl** command is described in more detail in the *Configuration and Command-Line Tool Reference*.

6.1.4.6.2. Initializing and Regenerating memberOf Attributes Using ldapmodify

Regenerating ***memberOf*** attributes is one of the tasks which can be managed through a special task configuration entry. Task entries occur under the **cn=tasks** configuration entry in the **dse.ldif** file, so it is also possible to initiate a task by adding the entry using **ldapmodify**. As soon as the task is complete, the entry is removed from the directory.

The **fixup-memberof.pl** script creates a special task entry in a Directory Server instance which regenerates the ***memberOf*** attributes.

To initiate a memberOf fixup task, add an entry under the **cn=memberof task, cn=tasks,cn=config** entry. The only required attribute is the ***cn*** for the specific task.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example memberOf,cn=memberof task,cn=tasks,cn=config
changetype: add
cn:example memberOf
```

As soon as the task is completed, the entry is deleted from the **dse.ldif** configuration, so it is possible to reuse the same task entry continually.

The **cn=memberof task** configuration is described in more detail in the *Configuration and Command-Line Tool Reference*.

6.1.5. Automatically Adding Entries to Specified Groups

Group management can be a critical factor for managing directory data, especially for clients which use Directory Server data and organization or which use groups to apply functionality to entries. Groups make it easier to apply policies consistently and reliably across the directory. Password policies, access control lists, and other rules can all be based on group membership.

Being able to assign new entries to groups, automatically, at the time that an account is created ensures that the appropriate policies and functionality are immediately applied to those entries — without requiring administrator intervention.

Dynamic groups are one method of creating groups and assigning members automatically because any matching entry is automatically included in the group. For applying Directory Server policies and settings, this is sufficient. However, LDAP applications and clients commonly need a static and explicit list of group members in order to perform whatever operation is required. And all of the members in static groups have to be manually added to those groups.

The static group itself cannot search for members like a dynamic group, but there is a way to allow a static group to have members added to it automatically — *the Auto Membership Plug-in*.

Automembership essentially allows a static group to act like a dynamic group, at least for adding new members to the group. Different automembership definitions create searches that are automatically run on all new directory entries. The automembership rules search for and identify matching entries — much like the dynamic search filters — and then explicitly add those entries as members to the static group.

NOTE

Automembership assignments are only made automatically when an entry is added to the Directory Server.

For existing entries or entries who are edited to match an automember group rule, there is a fix-up task which can be run that updates the group membership.

The Auto Membership Plug-in can target any type of object stored in the directory: users, machines and network devices, customer data, or other assets.

NOTE

The Auto Membership Plug-in adds a new entry to an existing group based on defined criteria. It does not create a group for the new entry.

To create a corresponding group entry when a new entry of a certain type is created, use the Managed Entries Plug-in. This is covered in Section 6.3, “Automatically Creating Dual Entries”.

6.1.5.1. Looking at the Structure of an Automembership Rule

The Auto Membership Plug-in itself is a container entry in **cn=plugins,cn=config**. Group assignments are defined through child entries.

6.1.5.1.1. The Automembership Configuration Entry

Automembership assignments are created through a main definition entry, a child of the Auto Membership Plug-in entry. Each definition entry defines three elements:

- An LDAP search to identify entries, including both a search scope and a search filter (***autoMemberScope*** and ***autoMemberFilter***)
- A default group to which to add the member entries (***autoMemberDefaultGroup***)
- The member entry format, which is the attribute in the group entry, such as ***member***, and the attribute value, such as ***dn*** (***autoMemberGroupingAttr***)

The definition is the basic configuration for an automember rule. It identifies all of the required information: what a matching member entry looks like and a group for that member to belong to.

For example, this definition assigns all Windows users to the **cn=windows-users** group:

```
dn: cn=Windows Users,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
autoMemberScope: ou=People,dc=example,dc=com
autoMemberFilter: objectclass=ntUser
autoMemberDefaultGroup: cn=windows-group,cn=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

The attributes for the definition entry are listed in Table 6.3, "Automember Definition Attributes".

Table 6.3. Automember Definition Attributes

Attribute	Description
autoMemberDefinition (required object class)	Identifies the entry as an automember definition. This entry must be a child of the Auto Membership Plug-in, **cn=Auto Membership Plugin,cn=plugins,cn=config**.
autoMemberScope	Sets the subtree DN to search for entries. This is the search base.
autoMemberFilter	Sets a standard LDAP search filter to use to search for matching entries. Examples of search filters are in Section 10.4, "LDAP Search Filters".
autoMemberDefaultGroup	Sets a default or fallback group to add the entry to as a member. If the definition does not use any regular expression conditions, then this is the primary group to which entries are added. If the automember definition does have defined regular expression conditions, then an entry is added to those specified groups first, and the ***autoMemberDefaultGroup*** group is used as a fallback for entries which match the ***autoMemberFilter*** but do not match a regular expression.
autoMemberGroupingAttr	Sets the name of the member attribute in the group entry and the attribute in the object entry that supplies the member attribute value. This structures how the Auto Membership Plug-in adds a member to the group, depending on the group configuration. For example, for a **groupOfUniqueNames** user group, each member is added as a ***uniqueMember*** attribute. The value of ***uniqueMember*** is the DN of the user entry. In essence, each group member is identified by the attribute-value pair of ***uniqueMember:*** *user_entry_DN*. The member entry format, then, is **uniqueMember:dn**.

6.1.5.1.2. Additional Regular Expression Entries

For something like a users group, where more than likely all matching entries should be added as members, a simple definition is sufficient. However, there can be instances where entries that match the LDAP search filter should be added to different groups, depending on the value of some other attribute. For example, machines may need to be added to different groups depending on their IP address or physical location; users may need to be in different groups depending on their employee ID number.

The automember definition can use regular expressions to provide additional conditions on what entries to include or exclude from a group, and then a new, specific group to add those selected entries to.

For example, an automember definition sets all machines to be added to a generic host group.

Example 6.2. Automember Definition for a Host Group

```
dn: cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: dc=example,dc=com
```

```
autoMemberFilter: objectclass=ipHost
autoMemberDefaultGroup: cn=systems,cn=hostgroups,ou=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

A regular expression rule is added so that any machine with a fully-qualified domain name within a given range is added to a web server group.

Example 6.3. Regular Expression Condition for a Web Server Group

```
dn: cn=webservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group for webservers
cn: webservers
autoMemberTargetGroup: cn=webservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^www\.web[0-9]+\.example\.com
```

So, any host machine added with a fully-qualified domain name that matches the expression **^www\.web[0-9]+\.example\.com**, such as **www.web1.example.com**, is added to the **cn=webservers** group, defined for that exact regular expression. Any other machine entry, which matches the LDAP filter **objectclass=ipHost** but with a different type of fully-qualified domain name, is added to the general host group, **cn=systems**, defined in the main definition entry.

The group in the definition, then, is a fallback for entries which match the general definition, but do not meet the conditions in the regular expression rule.

Regular expression rules are child entries of the automember definition.

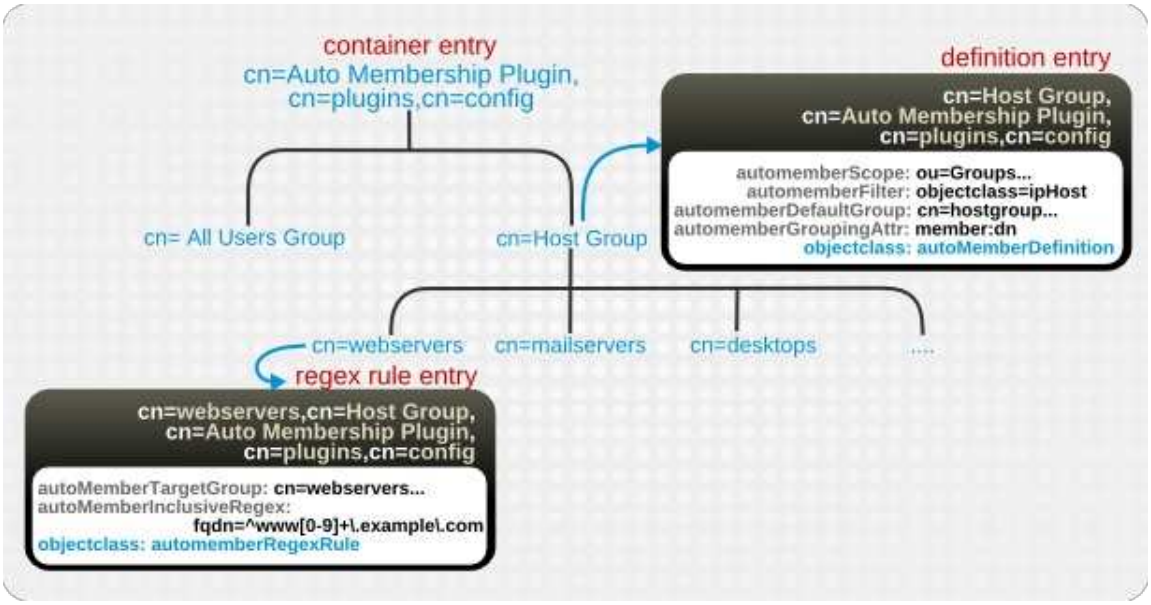


Figure 6.1. Regular Expression Conditions

Each rule can include multiple inclusion and exclusion expressions. (Exclusions are evaluated first.) If an entry matches any inclusion rule, it is added to the group.

There can be only one target group given for the regular expression rule.

Table 6.4. Regular Expression Condition Attributes

Attribute	Description
autoMemberRegexRule (required object class)	Identifies the entry as a regular expression rule. This entry must be a child of an automember definition (**objectclass: autoMemberDefinition**).

Attribute	Description
autoMemberInclusiveRegex	Sets a regular expression to use to identify entries to include. Only matching entries are added to the group. Multiple regular expressions could be used, and if an entry matches any one of those expressions, it is included in the group. The format of the expression is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see the pcresyntax(3) man page. This is a multi-valued attribute.
autoMemberExclusiveRegex	Sets a regular expression to use to identify entries to exclude. If an entry matches the exclusion condition, then it is *not* included in the group. Multiple regular expressions could be used, and if an entry matches any one of those expressions, it is excluded in the group. The format of the expression is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see the pcresyntax(3) man page. This is a multi-valued attribute.  **NOTE** Exclude conditions are evaluated first and take precedence over include conditions.
autoMemberTargetGroup	Sets which group to add the entry to as a member, if it meets the regular expression conditions.

6.1.5.2. Examples of Automembership Rules

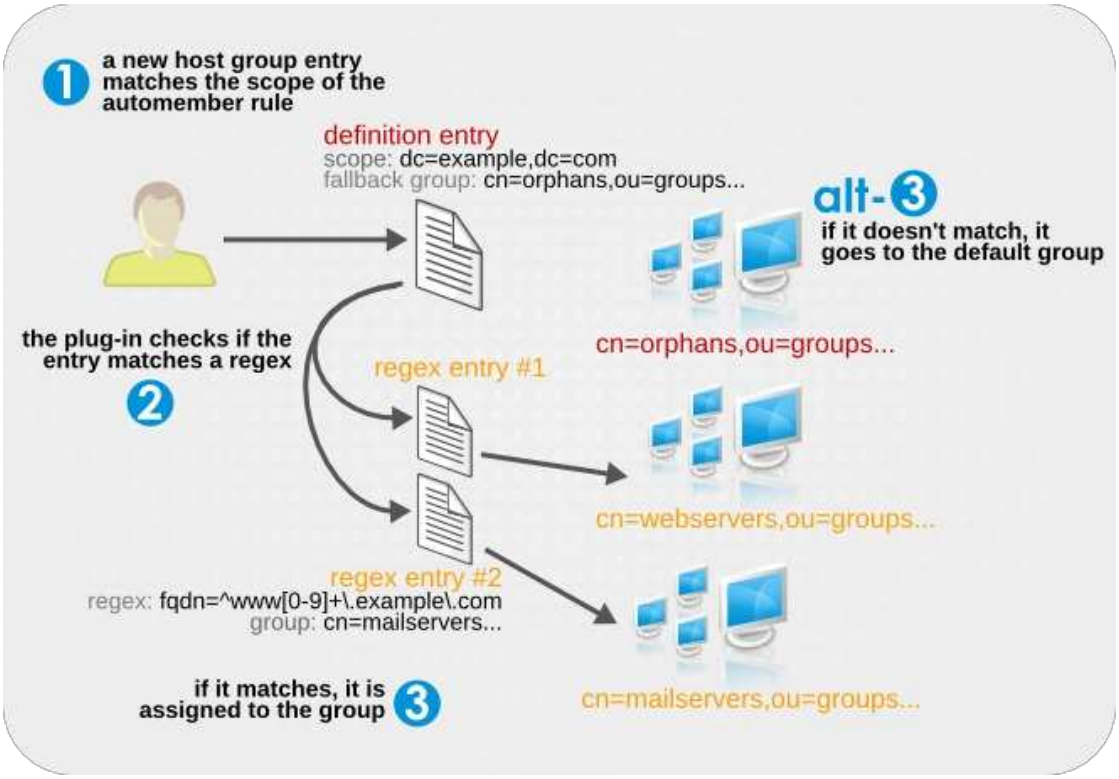
Automembership rules are usually going to applied to users and to machines (although they can be applied to any type of entry). There are a handful of examples that may be useful in planning automembership rules:

- Different host groups based on IP address
- Windows user groups
- Different user groups based on employee ID

Example 6.4. Host Groups by IP Address

The automember rule first defines the scope and target of the rule. The example in Section 6.1.5.1.2, "Additional Regular Expression Entries" uses the configuration group to define the fallback group and a regular expression entry to sort out matching entries.

The scope is used to find *all* host entries. The plug-in then iterates through the regular expression entries. If an entry matches an inclusive regular expression, then it is added to that host group. If it does not match any group, it is added to the default group.



The actual plug-in configuration entries are configured like this, for the definition entry and two regular expression entries to filter hosts into a web servers group or a mail servers group.

```
configuration entry
dn: cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: dc=example,dc=com
autoMemberFilter: objectclass=bootableDevice
autoMemberDefaultGroup: cn=orphans,cn=hostgroups,dc=example,dc=com
autoMemberGroupingAttr: member:dn

regex entry #1
dn: cn=webservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group placement for webservers
cn: webservers
autoMemberTargetGroup: cn=webservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^www[0-9]+\.example\.com
autoMemberInclusiveRegex: fqdn=^web[0-9]+\.example\.com
autoMemberExclusiveRegex: fqdn=^www13\.example\.com
autoMemberExclusiveRegex: fqdn=^web13\.example\.com

regex entry #2
dn: cn=mailservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group placement for mailservers
cn: mailservers
autoMemberTargetGroup: cn=mailservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^mail[0-9]+\.example\.com
autoMemberInclusiveRegex: fqdn=^smtp[0-9]+\.example\.com
autoMemberExclusiveRegex: fqdn=^mail13\.example\.com
autoMemberExclusiveRegex: fqdn=^smtp13\.example\.com
```

Example 6.5. Windows User Group

The basic users group shown in Section 6.1.5.1.1, "The Automembership Configuration Entry" uses the ***posixAccount*** attribute to identify all new users. All new users created within Directory Server are created with the ***posixAccount*** attribute, so that is a safe catch-all for new Directory Server users. However, when user accounts are synced over from the Windows domain to the Directory Server, the Windows user accounts are created *without* the ***posixAccount*** attribute.

Windows users are identified by the ***ntUser*** attribute. The basic, all-users group rule can be modified to target Windows users specifically, which can then be added to the default all-users group or to a Windows-specific group.

```
dn: cn=Windows Users,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
autoMemberScope: dc=example,dc=com
autoMemberFilter: objectclass=ntUser
autoMemberDefaultGroup: cn=Windows Users,cn=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

Example 6.6. User Groups by Employee Type

The Auto Membership Plug-in can work on custom attributes, which can be useful for entries which are managed by other applications. For example, a human resources application may create and then reference users based on the employee type, in a custom ***employeeType*** attribute.

Much like Example 6.4, "Host Groups by IP Address" , the user type rule uses two regular expression filters to sort full time and temporary employees, only this example uses an explicit value rather than a true regular expression. For other attributes, it may be more appropriate to use a regular expression, like basing the filter on an employee ID number range.

```
configuration entry
dn: cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: ou=employees,ou=people,dc=example,dc=com
autoMemberFilter: objectclass=inetorgperson
autoMemberDefaultGroup: cn=general,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn

regex entry #1
dn: cn=full time,cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group for full time employees
cn: full time
autoMemberTargetGroup: cn=full time,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberInclusiveRegex: employeeType=full

regex entry #2
dn: cn=temporary,cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group placement for interns, contractors, and seasonal employees
cn: temporary
autoMemberTargetGroup: cn=temporary,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberInclusiveRegex: employeeType=intern
autoMemberInclusiveRegex: employeeType=contractor
autoMemberInclusiveRegex: employeeType=seasonal
```

6.1.5.3. Creating Automembership Definitions

1. If necessary, enable the Auto Membership Plug-in.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Auto Membership Plugin,cn=plugins,cn=config
changetype: replace
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
-
```

2. Create the new plug-in instance below the **cn=Auto Membership Plugin,cn=plugins,cn=config** container entry. This entry must belong to the **autoMemberDefinition** object class.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Example Automember Definition,cn=Auto Membership Plugin,cn=plugins,cn=config
```

```
objectclass: autoMemberDefinition
```

The required attributes for the definition are listed in Table 6.3, "Automember Definition Attributes".

3. Set the scope and filter for the definition. This is used for the initial search for matching entries.

   For example, for new entries added to the **ou=People** subtree and containing the ***ntUser*** attribute:

```
autoMemberScope: ou=People,dc=example,dc=com
autoMemberFilter: objectclass=ntUser
```

4. Set the group to which to add matching entries (as the default or fallback group) and the format of the member entries for that group type.

```
autoMemberDefaultGroup: cn=windows-group,cn=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

5. *Optional.* Create inclusive or exclusive regular expression filters and set a group to use for entries matching those filters.

   The attributes for the regular expression condition are listed in Table 6.4, "Regular Expression Condition Attributes".

   Regular expression conditions are added as children of the automember definition. These conditions must belong to the **autoMemberRegexRule** object class.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Example Regex,cn=Example Automember Definition,cn=Auto Membership
Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
```

   Then add the target group name and any inclusive or exclusive regular expressions. Both include and exclude conditions can be used, and multiple expressions of both types can be used.

```
autoMemberTargetGroup: cn=webservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^www\.web[0-9]+\.example\.com
```

   If a new entry matches a regular expression condition, it is added to that group *instead of* the default group set in the automember definition.

6. Restart the Directory Server to load the modified new plug-in instance.

```
service dirsrv restart instance_name
```

6.1.5.4. Updating Existing Entries for Automembership Definitions

The Auto Member Plug-in only runs when new entries are added to the directory. The plug-in ignores existing entries or entries which are edited to match an automembership rule.

There is a directory task operation which can be run to check existing entries against automembership rules and then update group membership accordingly. This task (**cn=automember rebuild membership**) requires three elements to run, based on LDAP search parameters to identify which existing entries to process:

- The search filter
- The search scope
- The base DN from which to begin the search

The specific task run also needs a name.

The task entry can be created using **ldapmodify**; when the task completes, the entry is automatically removed. For example:

```
[root@server ~]# ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=my rebuild task, cn=automember rebuild membership,cn=tasks,cn=config
objectClass: top
```

```
objectClass: extensibleObject
cn: my rebuild task
basedn: dc=example,dc=com
filter: (uid=*)
scope: sub
```

6.1.5.5. Testing Automembership Definitions

Because each instance of the Auto Member Plug-in is a set of related-but-separate entries for the definition and regular expression, it can be difficult to see exactly how users are going to be mapped to groups. This becomes even more difficult when there are multiple rules which target different subsets of users.

There are two dry-run tasks which can be useful to determine whether all of the different Auto Member Plug-in definitions are assigning groups properly as designed.

Testing with Existing Entries

cn=automember export updates runs against *existing entries* in the directory and exports the results of what users would have been added to what groups, based on the rules. This is useful for testing existing rules against existing users to see how your real deployment are performing.

This task requires the same information as the **cn=automember rebuild membership** task — the base DN to search, search filter, and search scope — and has an additional parameter to specify an export LDIF file to record the proposed entry updates.

```
[root@server ~]# ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=test export, cn=automember export updates,cn=tasks,cn=config
objectClass: top
objectClass: extensibleObject
cn: test export
basedn: dc=example,dc=com
filter: (uid=*)
scope: sub
ldif: /tmp/automember-updates.ldif
```

Testing with an Import LDIF

cn=automember map updates takes *an import LDIF* of new users and then runs the new users against the current automembership rules. This can be very useful for testing a new rule, before applying it to (real) new or existing user entries.

This is called a map task because it maps or relates changes for proposed new entries to the existing rules.

This task only requires two attributes: the location of the input LDIF (which must contain at least some user entries) and an output LDIF file to which to write the proposed entry updates. Both the input and output LDIF files are absolute paths on the local machine.

For example:

```
[root@server ~]# ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=test mapping, cn=automember map updates,cn=tasks,cn=config
objectClass: top
objectClass: extensibleObject
cn: test mapping
ldif_in: /tmp/entries.ldif
ldif_out: /tmp/automember-updates.ldif
```

6.2. USING ROLES

Roles are an entry grouping mechanism that unify the static and dynamic groups described in the previous sections. Roles are designed to be more efficient and easier to use for applications. For example, an application can get the list of roles of which an entry is a member by querying the entry itself, rather than selecting a group and browsing the members list of several groups.

6.2.1. About Roles

Red Hat has two kinds of groups. *Static groups* have a finite and defined list of members. *Dynamic groups* used filters to recognize which entries are members of the group, so the group membership is constantly changed as the entries which match the group filter change. (Both kinds of groups are described in Section 6.1, “Using Groups”.)

Roles are a sort of hybrid group, behaving as both a static and a dynamic group. With a group, entries are added to a group entry as members. With a role, the role attribute is added to an entry and then that attribute is used to identify members in the role entry automatically.

Role *members* are entries that possess the role. Members can be specified either explicitly or dynamically. How role membership is specified depends upon the type of role. Directory Server supports three types of roles:

- *Managed roles* have an explicit enumerated list of members.
- *Filtered roles* are assigned entries to the role depending upon the attribute contained by each entry, specified in an LDAP filter. Entries that match the filter possess the role.
- *Nested roles* are roles that contain other roles.

Managed roles can do everything that can normally be done with static groups. The role members can be filtered using filtered roles, similarly to the filtering with dynamic groups. Roles are easier to use than groups, more flexible in their implementation, and reduce client complexity.

When a role is created, determine whether a user can add themselves or remove themselves from the role. See Section 6.2.10, "Using Roles Securely" for more information about roles and access control.

NOTE

Evaluating roles is more resource-intensive for the Directory Server than evaluating groups because the server does the work for the client application. With roles, the client application can check role membership by searching for the ***nsRole*** attribute. The ***nsRole*** attribute is a computed attribute, which identifies to which roles an entry belongs; the ***nsRole*** attribute is not stored with the entry itself. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

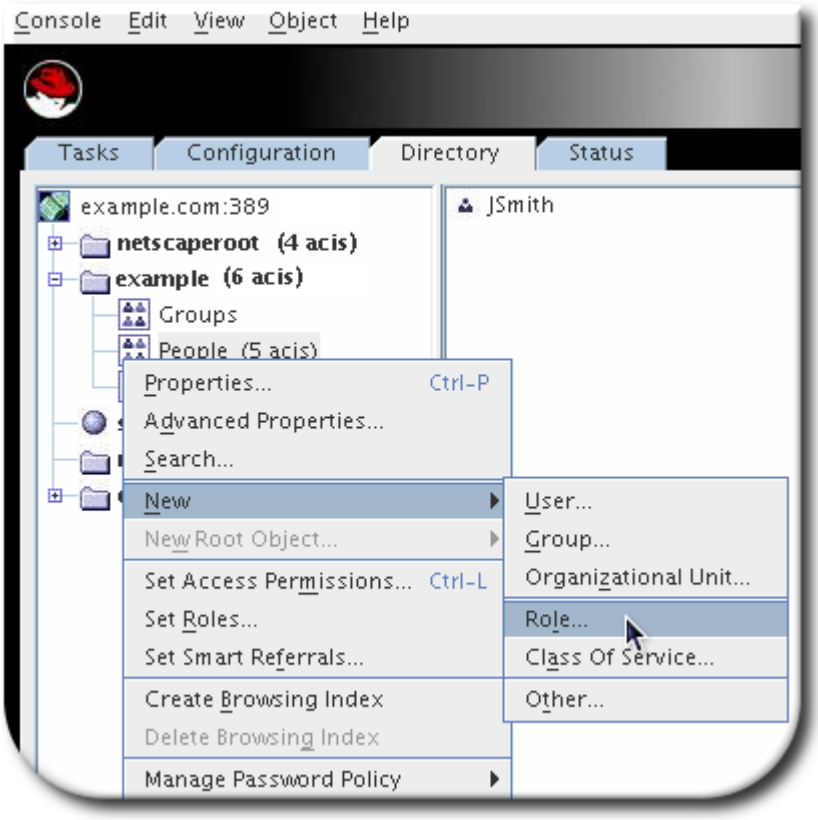
Considerations for using roles are covered in the *Directory Server Deployment Guide*.

6.2.2. Creating a Managed Role

Managed roles have an explicit enumerated list of members. Managed roles are added to entries by adding the ***nsRoleDN*** attribute to the entry.

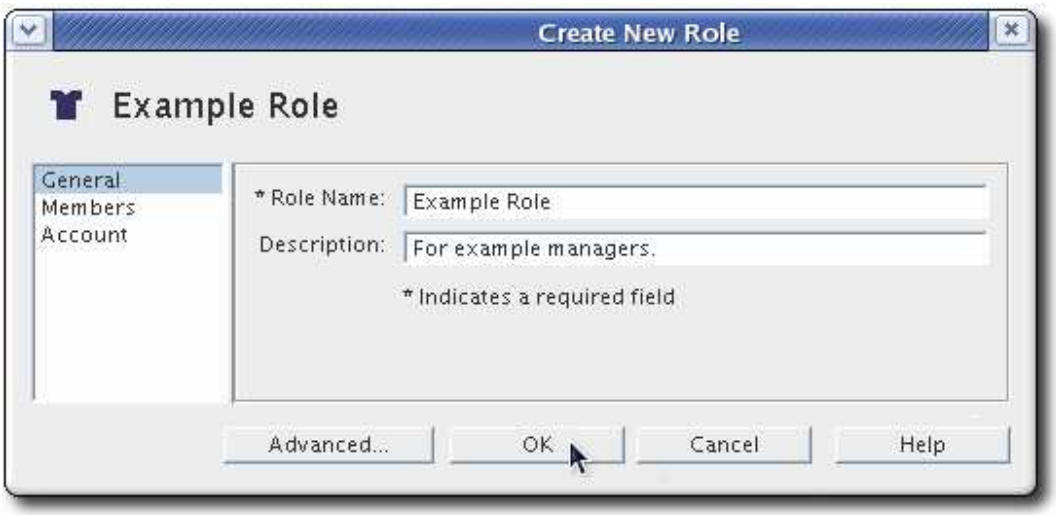
6.2.2.1. Creating a Managed Role in the Console

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new role.
3. Go to the **Object** menu, and select **New > Role**.



Alternatively, right-click the entry and select **New > Role**.

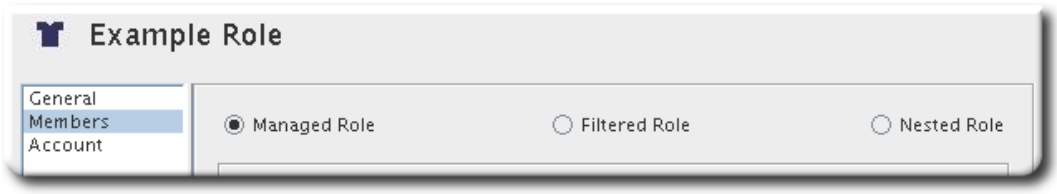
4. Click **General** in the left pane. Type a name for the new role in the **Role Name** field. The role name is required.



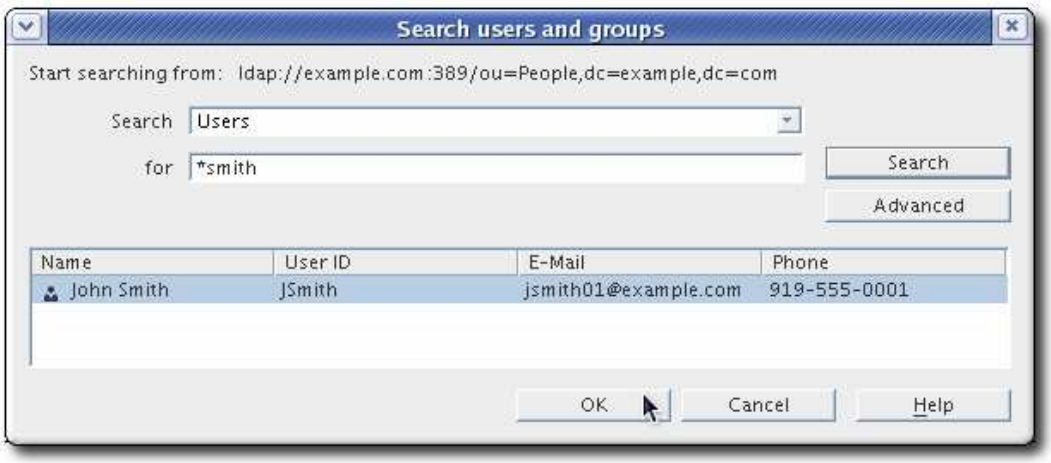
5. Enter a description of the new role in the **Description** field.

6. Click **Members** in the left pane.

7. In the right pane, select Managed Role. Click **Add** to add new entries to the list of members.



8. In the **Search** drop-down list, select **Users** from the **Search** drop-down list, then click **Search**. Select one of the entries returned, and click **OK**.



9. After adding all of the entries, click **OK**.

6.2.2.2. Creating Managed Roles through the Command Line

Roles inherit from the **ldapsubentry** object class, which is defined in the ITU X.509 standard. In addition, each managed role requires two object classes that inherit from the **nsRoleDefinition** object class:

- nsSimpleRoleDefinition
- nsManagedRoleDefinition

A managed role also allows an optional ***description*** attribute.

Members of a managed role have the ***nsRoleDN*** attribute in their entry.

This example creates a role which can be assigned to the marketing department.

1. Use **ldapmodify** with the **-a** option to add the managed role entry. The new entry must contain the **nsManagedRoleDefinition** object class, which in turn inherits from the **LdapSubEntry**, **nsRoleDefinition**, and **nsSimpleRoleDefinition** object classes.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Marketing,ou=people,dc=example,dc=com
objectclass: top
objectclass: LdapSubEntry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: Marketing
description: managed role for marketing staff
```

2. Assign the role to the marketing staff members, one by one, using **ldapmodify**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Bob,ou=people,dc=example,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
```

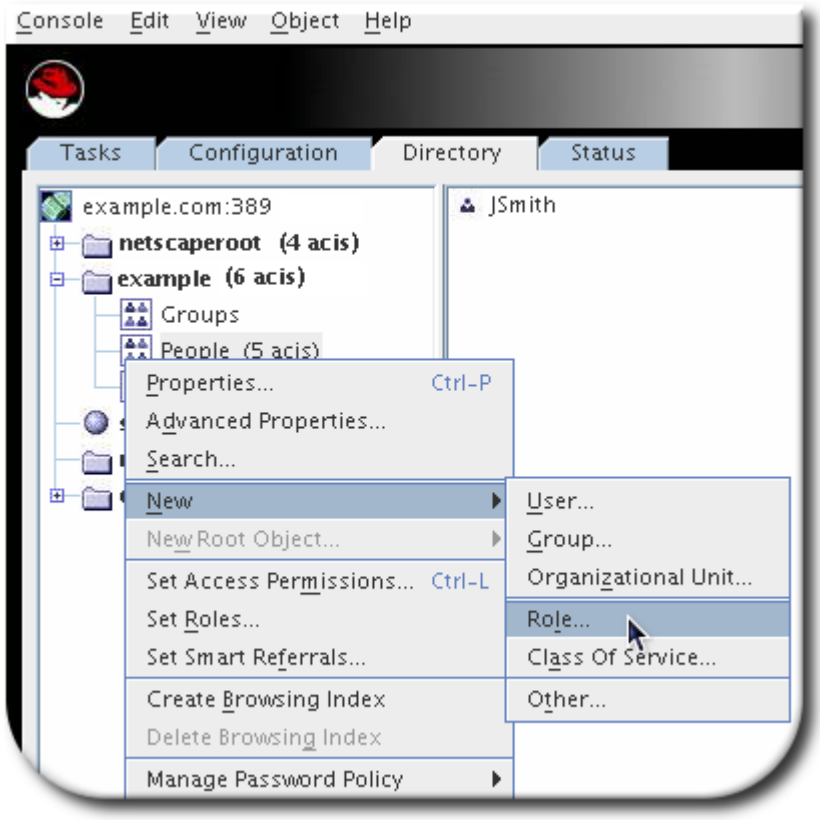
   The ***nsRoleDN*** attribute in the entry indicates that the entry is a member of a managed role, **cn=Marketing,ou=people,dc=example,dc=com**.

6.2.3. Creating a Filtered Role

Entries are assigned to a filtered role depending whether the entry possesses a specific attribute defined in the role. The role definition specifies an LDAP filter for the target attributes. Entries that match the filter possess (are members of) the role.

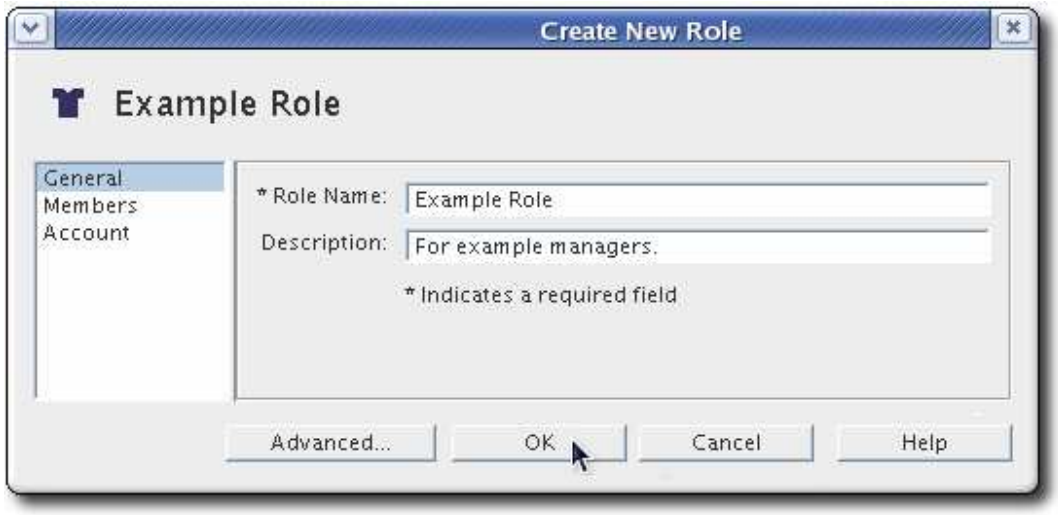
6.2.3.1. Creating a Filtered Role in the Console

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new role.
3. Go to the **Object** menu, and select **New > Role**.



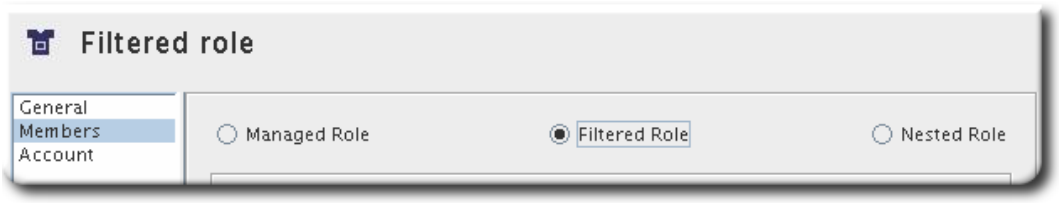
Alternatively, right-click the entry and select **New > Role**.

4. Click **General** in the left pane. Type a name for the new role in the **Role Name** field. The role name is required.



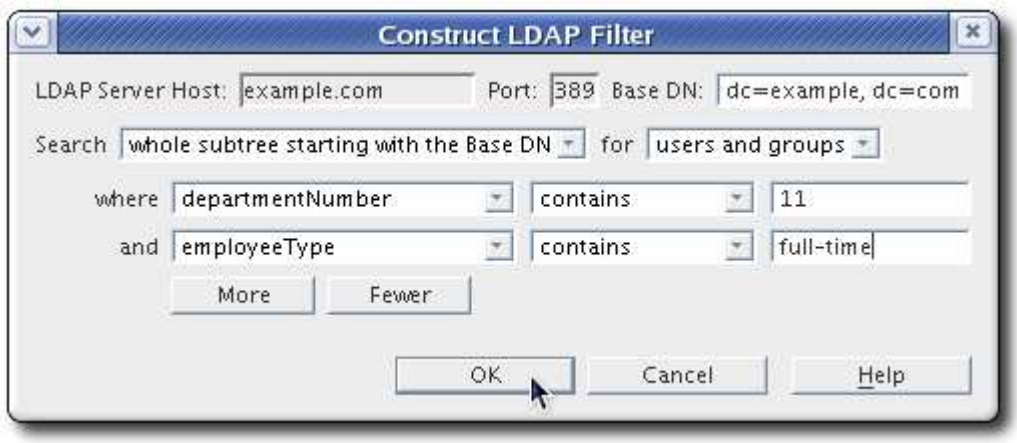
5. Enter a description of the new role in the **Description** field.
6. Click **Members** in the left pane.

   A search dialog box appears briefly.
7. In the right pane, select **Filtered Role**.



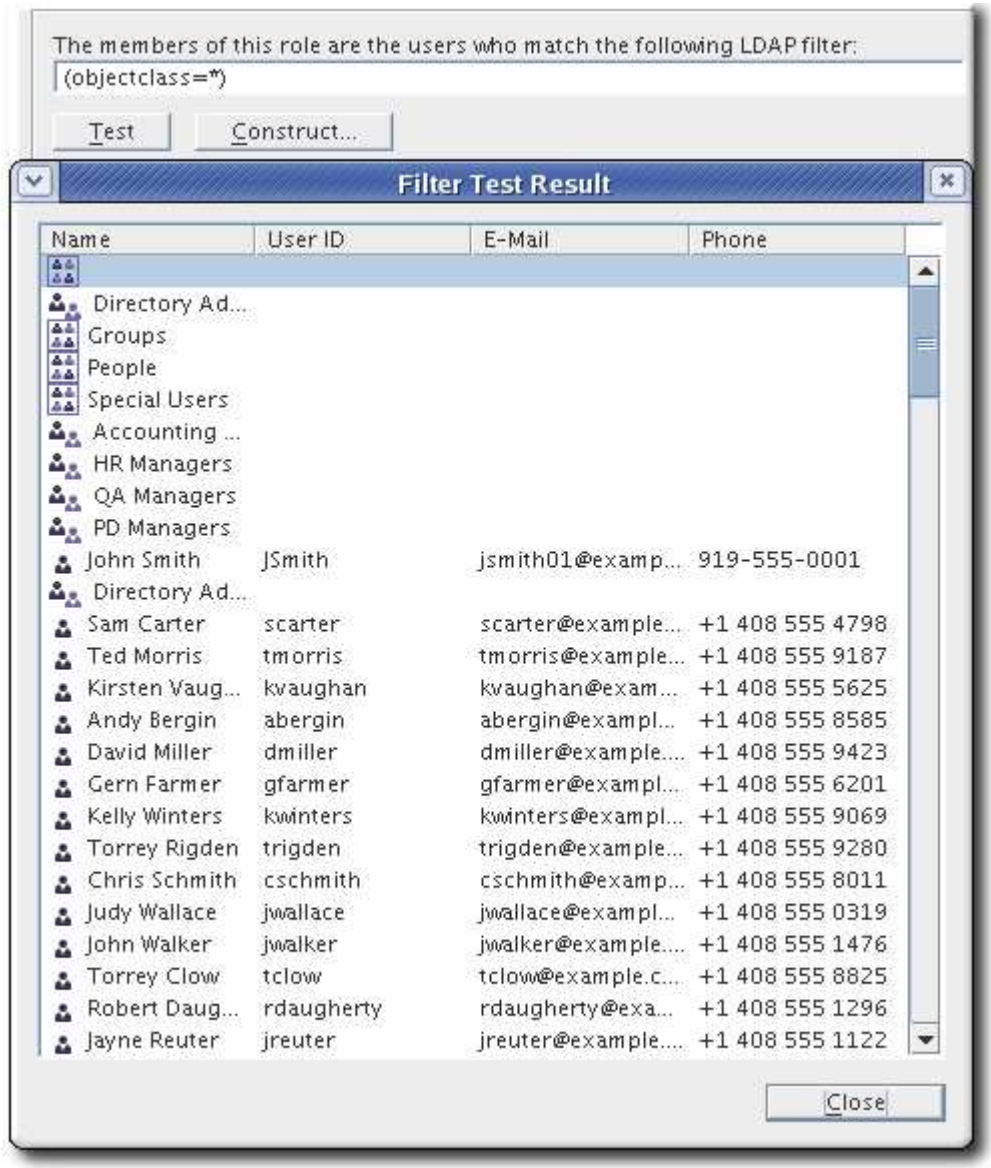
8. Enter an LDAP filter in the text field, or click **Construct** to be guided through the construction of an LDAP filter.

   The **Construct** opens the standard LDAP URL construction dialog. Ignore the fields for **LDAP Server Host**, **Port**, **Base DN**, and **Search** (since the search scope cannot be set filtered role definitions).



   - Select the types of entries to filter from the **For** drop-down list. The entries can be users, groups, or both.
   - Select an attribute from the **Where** drop-down list. The two fields following it refine the search by selecting one of the qualifiers from the drop-down list, such **as contains**, **does not contain**, **is**, or **is not**. Enter an attribute value in the text box. To add additional filters, click **More**. To remove unnecessary filters, click **Fewer**.

9. Click **Test** to try the filter.



10. Click **OK**.

6.2.3.2. Creating a Filtered Role through the Command Line

Roles inherit from the **ldapsubentry** object class, which is defined in the ITU X.509 standard. In addition, each filtered role requires two object classes that inherit from the **nsRoleDefinition** object class:

- nsComplexRoleDefinition
- nsFilteredRoleDefinition

A filtered role entry also requires the ***nsRoleFilter*** attribute to define the LDAP filter to determine role members. Optionally, the role can take a ***description*** attribute.

Members of a filtered role are entries that match the filter specified in the ***nsRoleFilter*** attribute.

This example creates a filtered role which is applied to all sales managers.

1. Run **ldapmodify** with the **-a** option to add a new entry:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Create the filtered role entry.

   The role entry has the **nsFilteredRoleDefinition** object class, which inherits from the **LdapSubEntry**, **nsRoleDefinition**, and **nsComplexRoleDefinition** object classes.

The **nsRoleFilter** attribute sets a filter for ***o*** (organization) attributes that contain a value of **sales managers**.

```
dn: cn=SalesManagerFilter,ou=people,dc=example,dc=com
changetype: add
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: SalesManagerFilter
nsRoleFilter: o=sales managers
Description: filtered role for sales managers
```

The following entry matches the filter (possesses the ***o*** attribute with the value **sales managers**), and, therefore, it is a member of this filtered role automatically:

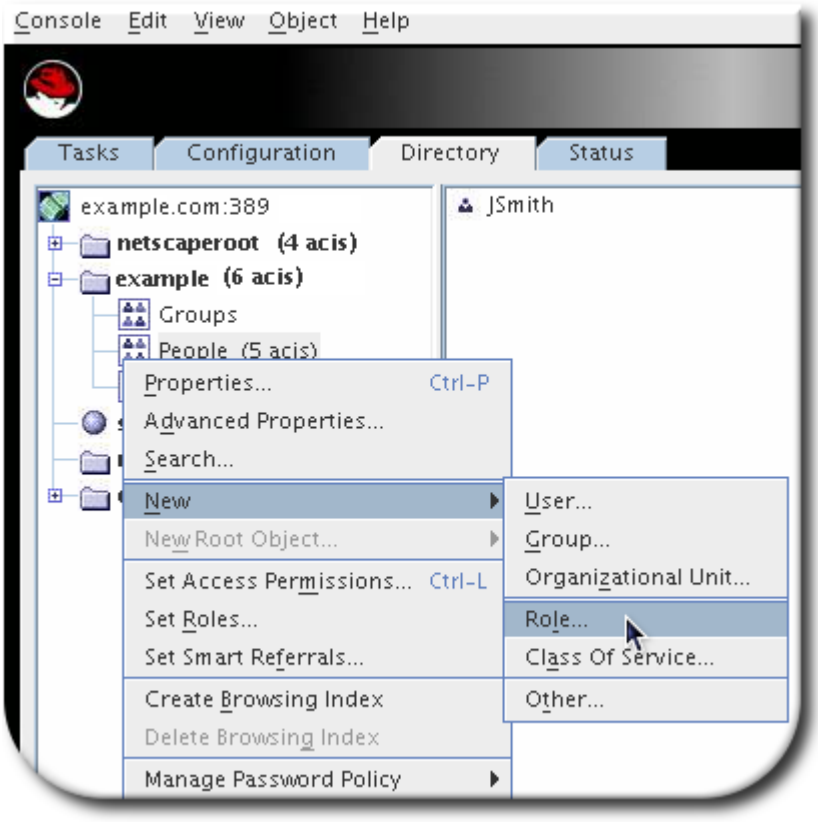
```
dn: cn=Pat Smith,ou=people,dc=example,dc=com
objectclass: person
cn: Pat
sn: Smith
userPassword: secret
o: sales managers
```

6.2.4. Creating a Nested Role

Nested roles are roles that contain other roles. Before it is possible to create a nested role, another role must exist. When a nested role is created, the Console displays a list of the roles available for nesting. The roles nested within the nested role are specified using the ***nsRoleDN*** attribute.

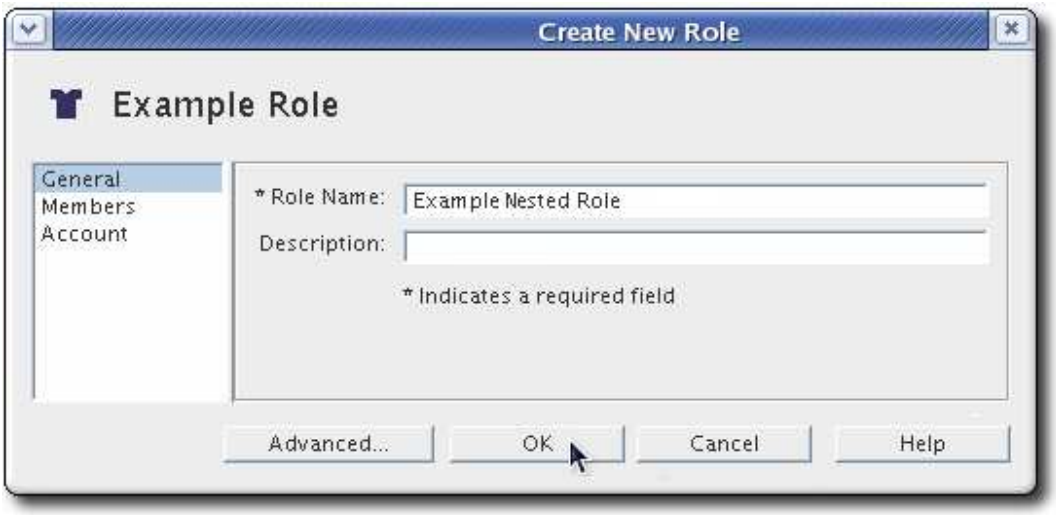
6.2.4.1. Creating a Nested Role in the Console

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new role.
3. Go to the **Object** menu, and select **New > Role**.



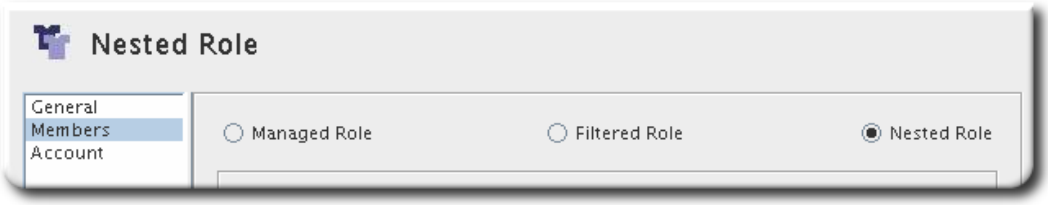
Alternatively, right-click the entry and select **New > Role**.

4. Click **General** in the left pane. Type a name for the new role in the **Role Name** field. The role name is required.



5. Click **Members** in the left pane.

6. In the right pane, select **Nested Role**.



7. Click **Add** to add roles to the list. The members of the nested role are members of other existing roles.

8. Select a role from the **Available roles** list, and click **OK**.

6.2.4.2. Creating Nested Role through the Command Line

Roles inherit from the **ldapsubentry** object class, which is defined in the ITU X.509 standard. In addition, each nested role requires two object classes that inherit from the **nsRoleDefinition** object class:

- nsComplexRoleDefinition
- nsNestedRoleDefinition

A nested role entry also requires the ***nsRoleDN*** attribute to identify the roles to nest within the container role. Optionally, the role can take a ***description*** attribute.

Members of a nested role are members of the roles specified in the ***nsRoleDN*** attributes of the nested role definition entry.

This example creates a single role out of the managed marketing role and filtered sales manager role.

1. Run **ldapmodify** with the **-a** option to add a new entry:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Create the nested role entry. The nested role has four object classes:

   - **nsNestedRoleDefinition**
   - **LDAPsubentry** (inherited)
   - **nsRoleDefinition** (inherited)
   - **nsComplexRoleDefinition** (inherited)

   The ***nsRoleDN*** attributes contain the DNs for both the marketing managed role and the sales managers filtered role.

```
dn: cn=MarketingSales,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsNestedRoleDefinition
cn: MarketingSales
nsRoleDN: cn=SalesManagerFilter,ou=people,dc=example,dc=com
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
```

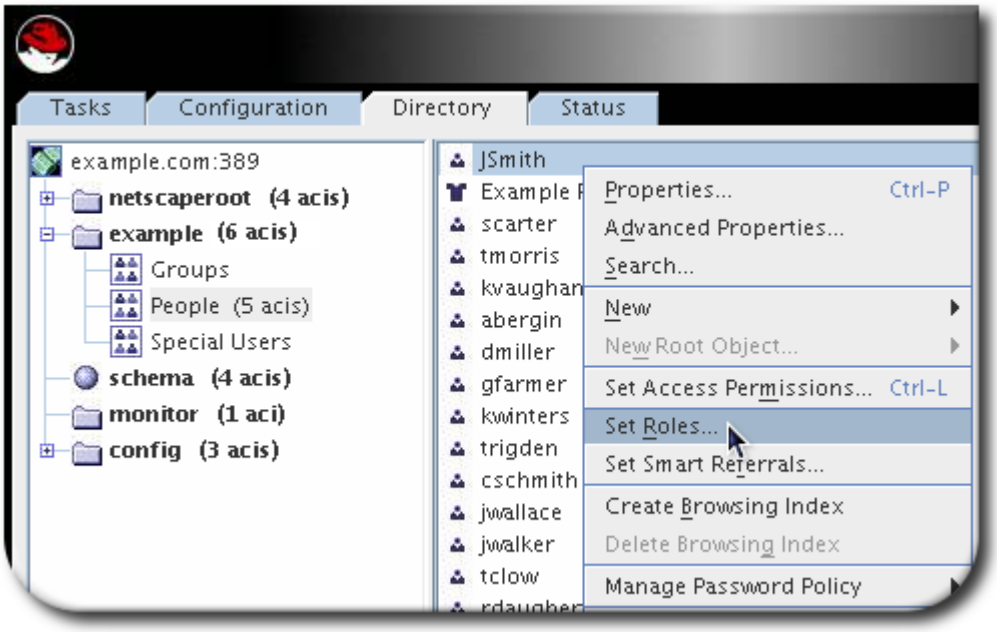
Both of the users in the previous examples, Bob and Pat, are members of this new nested role.

6.2.5. Editing and Assigning Roles to an Entry

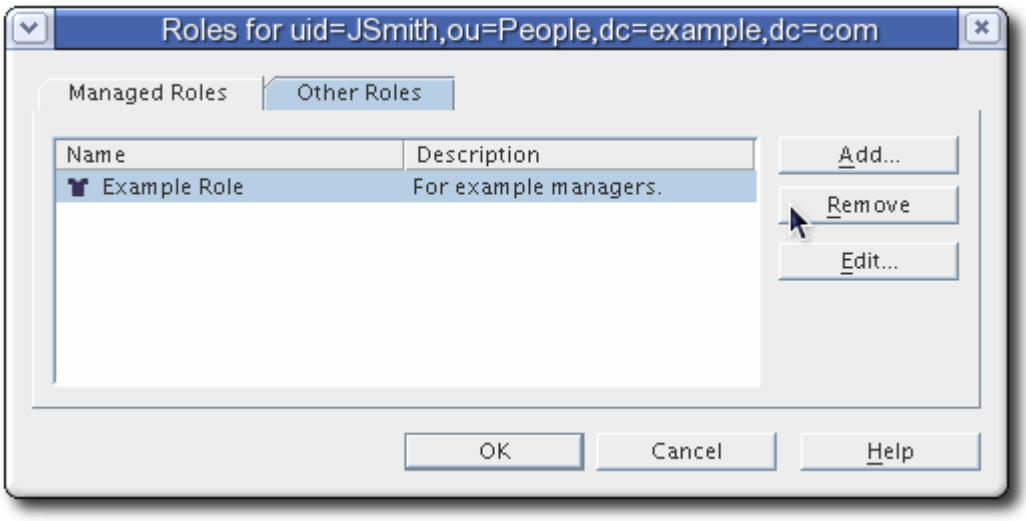
The entries which belong to a role are assigned on the role entry itself. For managed roles, user entries are added explicitly; for filtered roles, they are added through the results of an LDAP filter.

User entries are assigned to the role through the command line by editing the role entry, either by adding the entry as a member or adjusting the filter so it is included. In the Directory Server Console, however, there is a shortcut to add entries to a role by apparently editing the desired user entry (but, functionally, this really edits the role entry).

1. Select the **Directory** tab.
2. In the left navigation pane, browse the tree, and select the entry for which to view or edit a role.
3. Select **Set Roles** from the **Object** menu.



4. Select the **Managed Roles** tab to display the managed roles to which this entry belongs.

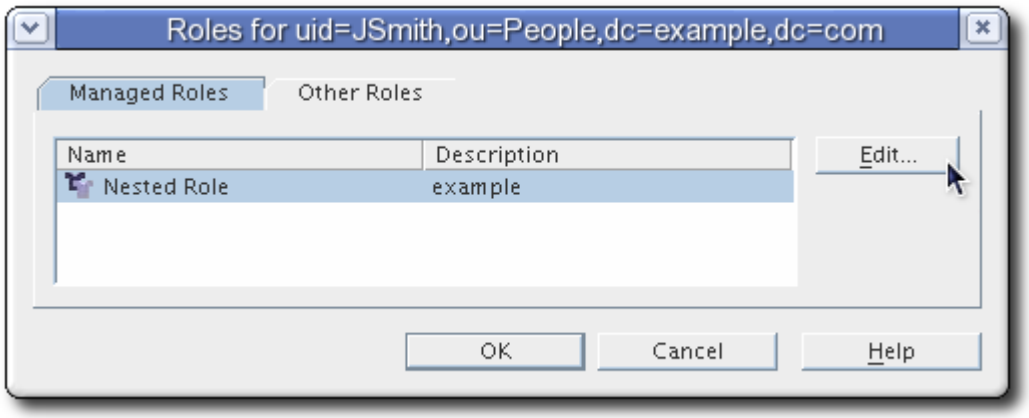


5. To add a new managed role, click **Add**, and select an available role from the **Role Selector** window.

NOTE

The configuration for a managed role associated with an entry can be edited by clicking the **Edit** button. The **Edit Entry** dialog box opens, and the general information or members for the role can be changed.

6. Select the **Other Roles** tab to view the filtered or nested roles to which this entry belongs.



Click **Edit** to make changes to any filtered or nested roles associated with the entry.

6.2.6. Viewing Roles for an Entry through the Command Line

Role assignments are always visible for an entry when it is displayed in the Directory Server Console. Role assignments are not returned automatically through the command line, however.

The nsRole attribute is an operational attribute. In LDAP, operational attributes must be requested explicitly in the search attributes list; they are not returned by default with the regular attributes in the schema of the entry. For example, this **ldapsearch** command returns the list of roles of which **uid=scarter** is a member, in addition to the regular attributes for the entry:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
“(uid=scarter)” \* nsRole

dn: uid=scarter,ou=people,dc=example,dc=com
objectClass: inetorgperson
objectClass: top
objectClass: person
objectClass: organizationalPerson
uid: scarter
cn: Sam Carter
sn: Carter
```

```
givenName: Sam
mail: scarter@example.com
userPassword: {SSHA}6BE31mhTfcYyIQF60kWlnEL8sIvPZ59hvFTRKw==
manager: uid=lbrown,ou=people,dc=example,dc=com
nsRole: cn=Role for Managers,dc=example,dc=com
nsRole: cn=Role for Accounting,dc=example,dc=com
```

IMPORTANT

Be sure to use the ***nsRole*** attribute, not the ***nsRoleDN*** attribute, to evaluate role membership.

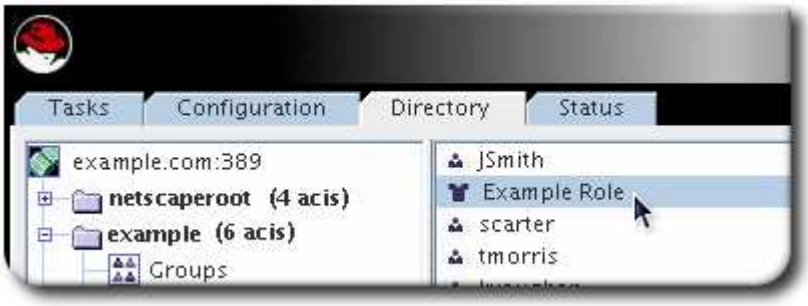
6.2.7. Making a Role Inactive or Active

The concept of activating/inactivating roles allows entire groups of entries to be activated or inactivated in just one operation. That is, the members of a role can be temporarily disabled by inactivating the role to which they belong.

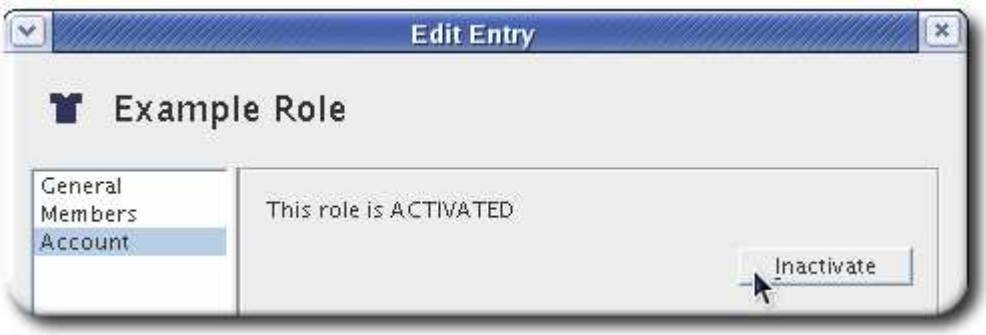
When a role is inactivated, it does not mean that the user cannot bind to the server using that role entry. The meaning of an inactivated role is that the user cannot bind to the server using any of the entries that belong to that role; the entries that belong to an inactivated role will have the ***nsAccountLock*** attribute set to **true**.

Members of a role can be temporarily disabled by inactivating the role to which they belong. Inactivating a role inactivates the entries possessed by the role, not the role itself.

1. Select the **Directory** tab.
2. Browse the navigation tree in the left pane to locate the base DN for the role. Roles appear in the right pane with other entries.



3. Double-click the role, open the **Account** tab, and click the **Inactivate** button.



Alternatively, select the role. Right-click the role and select **Inactivate** from the menu.

The role is inactivated.

To reactivate a disabled role, re-open the role configuration or open the **Object** menu, and select **Activate**. All of the members of the role are re-enabled.

6.2.8. Viewing the Activation Status for Entries

When a nested role is inactivated, a user cannot bind to the server if it is a member of any role within the nested role. All the entries that belong to a role that directly or indirectly are members of the nested role have ***nsAccountLock*** set to **true**. There can be several layers of nested roles, and inactivating a nested role at any point in the nesting will inactivate all roles and users beneath it.

The Directory Server Console automatically shows the active or inactive status of entries.

To see the inactivated entries, select **Inactivation State** from the **View** menu. Members of an inactivated role have a red slash through them. For example, John Smith is a member of the inactive Example Role.

The ***nsAccountLock*** attribute is an operational attribute and must be explicitly requested in the search command in the list of search attributes. For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
“(uid=scarter)” nsAccountLock
```

6.2.9. About Deleting Roles

Deleting a role deletes the role entry but does not delete the ***nsRoleDN*** attribute for each role member. To delete the ***nsRoleDN*** attribute for each role member, enable the Referential Integrity plug-in, and configure it to manage the ***nsRoleDN*** attribute. For more information on the Referential Integrity plug-in, see Section 3.6, “Maintaining Referential Integrity”.

6.2.10. Using Roles Securely

Not every role is suitable for use in a security context. When creating a new role, consider how easily the role can be assigned to and removed from an entry. Sometimes it is appropriate for users to be able to add or remove themselves easily from a role. For example, if there is an interest group role called **Mountain Biking**, interested users should be able to add themselves or remove themselves easily.

However, it is inappropriate to have such open roles for some security situations. One potential security risk is inactivating user accounts by inactivating roles. Inactive roles have special ACIs defined for their suffix. If an administrator allows users to add and remove themselves from roles freely, then in some circumstance, they may be able to remove themselves from an inactive role to prevent their accounts from being locked.

For example, user A possesses the managed role, **MR**. The **MR** role has been locked using account inactivation. This means that user A cannot bind to the server because the ***nsAccountLock*** attribute is computed as **true** for that user. However, if user A was already bound to Directory Server and noticed that he is now locked through the MR role, the user can remove the ***nsRoleDN*** attribute from his entry and unlock himself if there are no ACIs preventing him.

To prevent users from removing the ***nsRoleDN*** attribute, use the following ACIs depending upon the type of role being used.

- *Managed roles.* For entries that are members of a managed role, use the following ACI to prevent users from unlocking themselves by removing the appropriate ***nsRoleDN***:

  ```
  aci: (targetattr="nsRoleDN") (targetattrfilters= add=nsRoleDN:(!
  (nsRoleDN=cn=AdministratorRole,dc=example,dc=com)), del=nsRoleDN:(!
  (nsRoleDN=cn=nsManagedDisabledRole,dc=example,dc=com))) (version3.0;acl "allow mod of nsRoleDN
  by self but not to critical values"; allow(write) userdn=ldap:///self;)
  ```

- *Filtered roles.* The attributes that are part of the filter should be protected so that the user cannot relinquish the filtered role by modifying an attribute. The user should not be allowed to add, delete, or modify the attribute used by the filtered role. If the value of the filter attribute is computed, then all attributes that can modify the value of the filter attribute should be protected in the same way.

- *Nested roles.* A nested role is comprised of filtered and managed roles, so both ACIs should be considered for modifying the attributes (***nsRoleDN*** or something else) of the roles that comprise the nested role.

For more information about account inactivation, see Section 14.11, “Manually Inactivating Users and Roles” .

6.3. AUTOMATICALLY CREATING DUAL ENTRIES

Some clients and integration with Red Hat Directory Server require dual entries. For example, both Posix systems typically have a group for each user. The Directory Server's *Managed Entries Plug-in* creates a new managed entry, with accurate and specific values for attributes, automatically whenever an appropriate origin entry is created.

6.3.1. About Managed Entries

The basic idea behind the Managed Entries Plug-in is that there are situations when Entry A is created and there should automatically be an Entry B with related attribute values. For example, when a Posix user (**posixAccount** entry) is created, a corresponding group entry (**posixGroup** entry) should also be created. An instance of the Managed Entries Plug-in identifies what entry (the *origin entry*) triggers the plug-in to automatically generate a new entry (the *managed entry*).

The plug-in works within a defined scope of the directory tree, so only entries within that subtree and that match the given search filter trigger a Managed Entries operation.

Much like configuring a class of service, a managed entry is configured through two entries:

- A definition entry, that identifies the scope of the plug-in instance and the template to use
- A template entry, that models what the final managed entry will look like

6.3.1.1. About the Instance Definition Entry

As with the Linked Attributes and DNA Plug-ins, the Managed Entries Plug-in has a container entry in **cn=plugins,cn=config**, and each unique configuration instance of the plug-in has a definition entry beneath that container.

An instance of the Managed Entries Plug-in defines three things:

- The search criteria to identify the origin entries (using a search scope and a search filter)
- The subtree under which to create the managed entries (the new entry location)
- The template entry to use for the managed entries

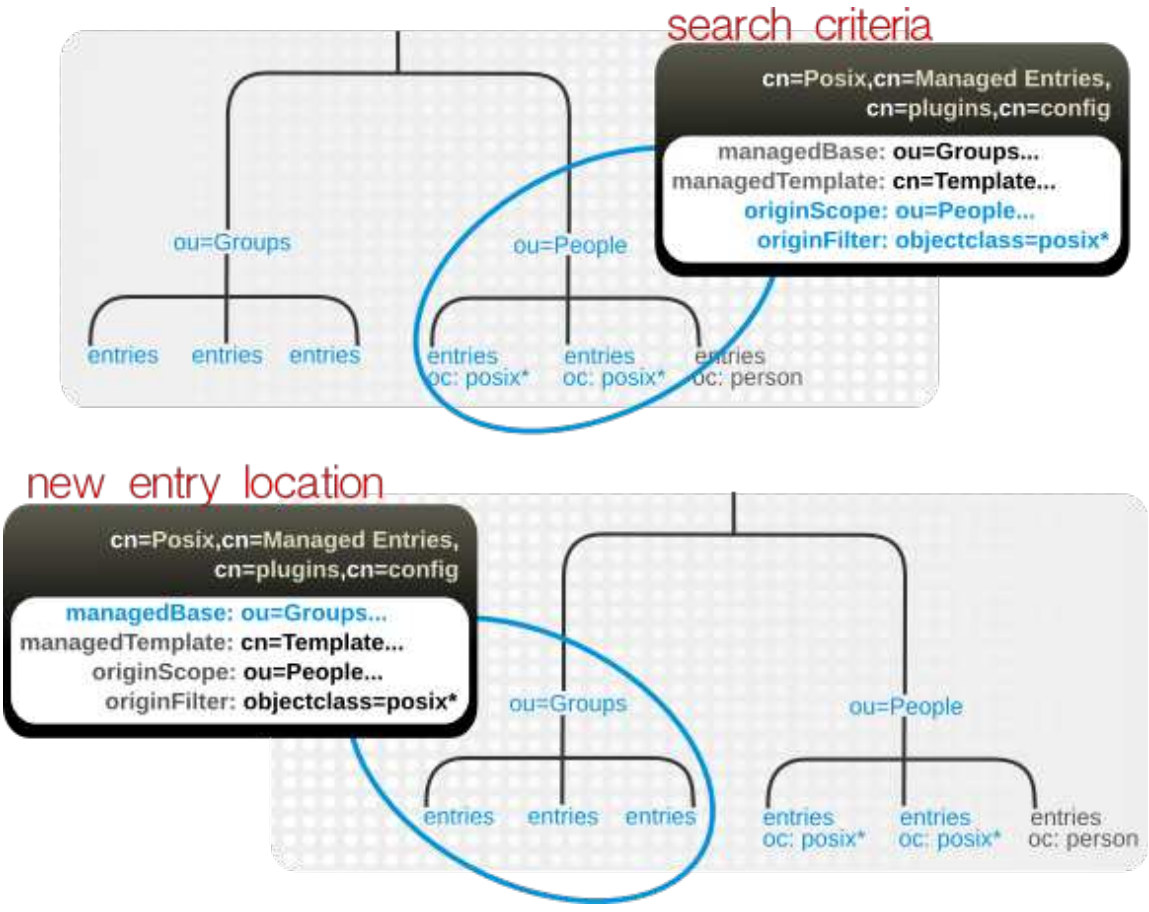


Figure 6.2. Defining Managed Entries

For example:

```
dn: cn=Posix User-Group,cn=Managed Entries,cn=plugins,cn=config
objectclass: extensibleObject
cn: Posix User-Group
originScope: ou=people,dc=example,dc=com
```

```
originFilter: objectclass=posixAccount
managedBase: ou=groups,dc=example,dc=com
managedTemplate: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
```

The origin entry does not have to have any special configuration or settings to create a managed entry; it simply has to be created within the scope of the plug-in and match the given search filter.

6.3.1.2. About the Template Entry

Each instance of the plug-in uses a template entry which defines the managed entry configuration. The template effectively lays out the entry, from the object classes to the entry values.

NOTE

Since the template is referenced in the definition entry, it can be located anywhere in the directory. However, it is recommended that the template entry be under the replicated suffix so that any other masters in multi-master replication all use the same template for their local instances of the Managed Entries Plug-in.

The concept of a template entry is similar to the templates used in CoS, but there are some important differences. The managed entry template is slightly different than the type of template used for a class of service. For a class of service, the template contains a single attribute with a specific value that is fed into all of the entries which belong to that CoS. Any changes to the class of service are immediately reflected in the associated entries, because the CoS attributes in those entries are virtual attributes, not truly attributes set on the entry.

The template entry for the Managed Entries Plug-in, on the other hand, is not a central entry that supplies values to associated entries. It is a true template – it lays out what is in the entry. The template entry can contain both static attributes (ones with pre-defined values, similar to a CoS) and mapped attributes (attributes that pull their values or parts of values from the origin entry). The template is referenced when the managed entry is created and then any changes are applied to the managed entry *only* when the origin entry is changed and the template is evaluated again by the plug-in to apply those updates.

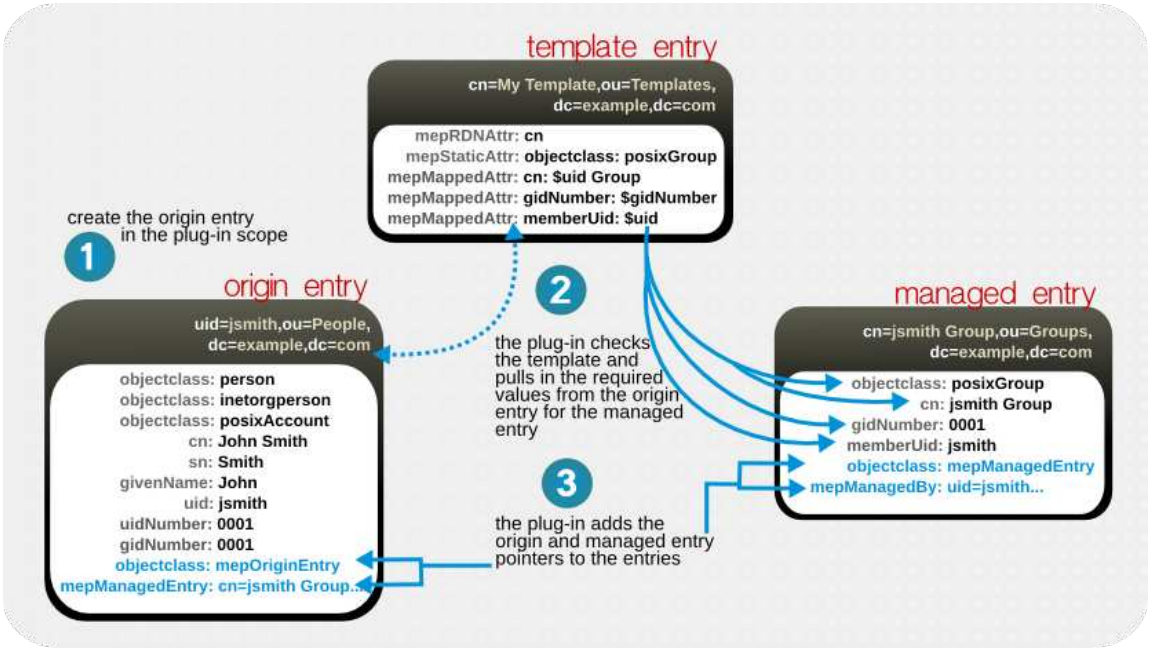


Figure 6.3. Templates, Managed Entries, and Origin Entries

The template can provide a specific value for an attribute in the managed entry by using a *static* attribute in the template. The template can also use a value that is derived from some attribute in the origin entry, so the value may be different from entry to entry; that is a *mapped* attribute, because it references the attribute type in the origin entry, not a value.

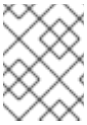
A mapped value use a combination of token (dynamic values) and static values, but it can only use *one token in a mapped attribute*.

```
dn: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
objectclass: mepTemplateEntry
cn: Posix User-Group Template
mepRDNAttr: cn
mepStaticAttr: objectclass: posixGroup
```

```
mepMappedAttr: cn: $cn Group Entry
mepMappedAttr: gidNumber: $gidNumber
mepMappedAttr: memberUid: $uid
```

The mapped attributes in the template use tokens, prepended by a dollar sign ($), to pull in values from the origin entry and use it in the managed entry. (If a dollar sign is actually in the managed attribute value, then the dollar sign can be escaped by using two dollar signs in a row.)

A mapped attribute definition can be quoted with curly braces, such as **Attr: ${cn}test**. Quoting a token value is not required if the token name is not immediately followed by a character that is valid in an attribute name, such as a space or comma. For example, **$cn test** is acceptable in an attribute definition because a space character immediately follow the attribute name, but **$cntest** is not valid because the Managed Entries Plug-in attempts to look for an attribute named **cntest** in the origin entry. Using curly braces identifies the attribute token name.



NOTE

Make sure that the values given for static and mapped attributes comply with the required attribute syntax.

6.3.1.3. Entry Attributes Written by the Managed Entries Plug-in

Both the origin entry and the managed entry have special managed entries attributes which indicate that they are being managed by an instance of the Managed Entries Plug-in. For the origin entry, the plug-in adds links to associated managed entries.

```
dn: uid=jsmith,ou=people,dc=example,dc=com
objectclass: mepOriginEntry
objectclass: posixAccount
...
sn: Smith
mail: jsmith@example.com
mepManagedEntry: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
```

On the managed entry, the plug-in adds attributes that point back to the origin entry, in addition to the attributes defined in the template.

```
dn: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
objectclass: mepManagedEntry
objectclass: posixGroup
...
mepManagedBy: uid=jsmith,ou=people,dc=example,dc=com
```

Using special attributes to indicate managed and origin entries makes it easy to identify the related entries and to assess changes made by the Managed Entries Plug-in.

6.3.1.4. Managed Entries Plug-in and Directory Server Operations

The Managed Entries Plug-in has some impact on how the Directory Server carries out common operations, like add and delete operations.

Table 6.5. Managed Entries Plug-in and Directory Server Operations

Operation	Effect by the Managed Entries Plug-in
Add	With every add operation, the server checks to see if the new entry is within the scope of any Managed Entries Plug-in instance. If it meets the criteria for an origin entry, then a managed entry is created and managed entry-related attributes are added to both the origin and managed entry.

Operation	Effect by the Managed Entries Plug-in
Modify	If an origin entry is modified, it triggers the plug-in to update the managed entry. Changing a *template* entry, however, does not update the managed entry automatically. Any changes to the template entry are not reflected in the managed entry until after the next time the origin entry is modified. The mapped managed attributes *within* a managed entry cannot be modified manually, only by the Managed Entry Plug-in. Other attributes in the managed entry (including static attributes added by the Managed Entry Plug-in) can be modified manually.
Delete	If an origin entry is deleted, then the Managed Entries Plug-in will also delete any managed entry associated with that entry. There are some limits on what entries can be deleted. ● A template entry cannot be deleted if it is currently referenced by a plug-in instance definition. ● A managed entry cannot be deleted except by the Managed Entries Plug-in.
Rename	If an origin entry is renamed, then plug-in updates the corresponding managed entry. If the entry is moved *out* of the plug-in scope, then the managed entry is deleted, while if an entry is moved *into* the plug-in scope, it is treated like an add operation and a new managed entry is created. As with delete operations, there are limits on what entries can be renamed or moved. ● A configuration definition entry cannot be moved out of the Managed Entries Plug-in container entry. If the entry is removed, that plug-in instance is inactivated. ● If an entry is moved *into* the Managed Entries Plug-in container entry, then it is validated and treated as an active configuration definition. ● A template entry cannot be renamed or moved if it is currently referenced by a plug-in instance definition. ● A managed entry cannot be renamed or moved except by the Managed Entries Plug-in.
Replication	The Managed Entries Plug-in operations *are not initiated by replication updates*. If an add or modify operation for an entry in the plug-in scope is replicated to another replica, that operation does not trigger the Managed Entries Plug-in instance on the replica to create or update an entry. The only way for updates for managed entries to be replicated is to replicate the final managed entry over to the replica.

6.3.2. Creating the Managed Entries Template Entry

The first entry to create is the template entry. The template entry must contain all of the configuration required for the generated, managed entry. This is done by setting setting the attribute-value assertions in static and mapped attributes in the template:

```
mepStaticAttr: attribute: specific_value
mepMappedAttr: attribute: $token_value
```

The static attributes set an explicit value; mapped attributes pull some value from the originating entry is used to supply the given attribute. The values of these attributes will be tokens in the form *attribute: $attr*. As long as the syntax of the expanded token of the attribute does not violate the required attribute syntax, then other terms and strings can be used in the attribute. For example:

```
mepMappedAttr: cn: Managed Group for $cn
```

There are some syntax rules that must be followed for the managed entries:

- A mapped value use a combination of token (dynamic values) and static values, but it can only use *one token per mapped attribute*.
- The mapped attributes in the template use tokens, prepended by a dollar sign ($), to pull in values from the origin entry and use it in the managed entry. (If a dollar sign is actually in the managed attribute value, then the dollar sign can be escaped by using two dollar signs in a row.)
- A mapped attribute definition can be quoted with curly braces, such as **Attr: ${cn}test**. Quoting a token value is not required if the token name is not immediately followed by a character that is valid in an attribute name, such as a space or comma. For example, **$cn test** is acceptable in an attribute definition because a space character immediately follow the attribute name, but **$cntest** is not valid because the Managed Entries Plug-in attempts to look for an attribute named **cntest** in the origin entry. Using curly braces identifies the attribute token name.
- Make sure that the values given for static and mapped attributes comply with the required attribute syntax.

NOTE

Make sure that the values given for static and mapped attributes comply with the required attribute syntax. For example, if one of the mapped attributes is ***gidNumber***, then the mapped value should be an integer.

Table 6.6. Attributes for the Managed Entry Template

Attribute	Description
mepTemplateEntry (object class)	Identifies the entry as a template.
cn	Gives the common name of the entry.
mepMappedAttr	Contains an attribute-token pair that the plug-in uses to create an attribute in the managed entry with a value taken from the originating entry.
mepRDNAttr	Specifies which attribute to use as the naming attribute in the managed entry. The attribute used as the RDN **must** be a mapped attribute for the configuration to be valid.
mepStaticAttr	Contains an attribute-value pair that will be used, with that specified value, in the managed entry.

To create a template entry:

1. Run **ldapmodify** to add the entry. This entry can be located anywhere in the directory tree.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Posix User Template,ou=templates,dc=example,dc=com
cn: Posix User Template
```

   You can also use the Directory Server Console to create the entry, as described in Section 3.1.2, “Creating Directory Entries”.

2. Give it the **mepTemplateEntry** object class to indicate that it is a template entry.

```
objectClass: top
objectclass: mepTemplateEntry
```

3. Set the attributes for the entry; these are described in Table 6.6, “Attributes for the Managed Entry Template”. The RDN attribute (***mepRDNAttr***) is required. The attribute parameters are optional and the values depend on the type of entry that the plug-in will create. Make sure that whatever attribute you use for the naming attribute is also contained in the template entry as a mapped attribute.

NOTE

Attributes which will be the same for each managed entry – like the object class for the entries – should use the ***mepStaticAttr*** attribute to set the values manually.

```
mepRDNAttr: cn
mepStaticAttr: objectclass: posixGroup
mepMappedAttr: cn: $cn Group Entry
mepMappedAttr: gidNumber: $gidNumber
mepMappedAttr: memberUid: $uid
```

6.3.3. Creating the Managed Entries Instance Definition

Once the template entry is created, then it is possible to create a definition entry that points to that template. The definition entry is an instance of the Managed Entries Plug-in.

NOTE

When the definition is created, the server checks to see if the specified template entry exists. If the template does not exist, then the server returns a warning that the definition configuration is invalid.

The definition entry must define the parameters to recognize the potential origin entry and the information to create the managed entry. The attributes available for the plug-in instance are listed in Table 6.7, "Attributes for the Managed Entries Definition Entry".

Table 6.7. Attributes for the Managed Entries Definition Entry

Attribute Name	Description
originFilter	The search filter to use to search for and identify the entries within the subtree which require a managed entry. The syntax is the same as a regular search filter.
originScope	The base subtree which contains the potential origin entries for the plug-in to monitor.
managedTemplate	Identifies the template entry to use to create the managed entry. This entry can be located anywhere in the directory tree.
managedBase	The subtree under which to create the managed entries.

NOTE

The Managed Entries Plug-in is enabled by default. If this plug-in is disabled, then re-enable it as described in Section 1.8.1, "Enabling Plug-ins in the Directory Server Console" .

To create an instance:

1. Create the new plug-in instance below the **cn=Managed Entries,cn=plugins,cn=config** container entry.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=instance_name,cn=Managed Entries,cn=plugins,cn=config
```

2. Set the scope and filter for the origin entry search, the location of the new managed entries, and the template entry to use. These required attributes are listed in Table 6.7, "Attributes for the Managed Entries Definition Entry".

```
objectClass: top
objectClass: extensibleObject
cn: Posix User-Group
originScope: ou=people,dc=example,dc=com
```

```
originFilter: objectclass=posixAccount
managedBase: ou=groups,dc=example,dc=com
managedTemplate: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
```

3. Restart the Directory Server to load the modified new plug-in instance.

```
service dirsrv restart instance_name
```

6.3.4. Putting Managed Entries Plug-in Configuration in a Replicated Database

As Section 6.3.1, "About Managed Entries" highlights, different instances of the Managed Entries Plug-in are created as children beneath the container plug-in entry in **cn=plugins,cn=com**. (This is common for plug-ins which allow multiple instances.) The drawback to this is that the configuration entries in **cn=plugins,cn=com** are not replicated, so the configuration has to be re-created on each Directory Server instance.

The Managed Entries Plug-in entry allows the ***nsslapd-pluginConfigArea*** attribute. This attribute to another container entry, in the main database area, which contains the plug-in instance entries. This container entry can be in a replicated database, which allows the plug-in configuration to be replicated.

1. Create a container entry in a subtree that is replicated.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x -D "cn=directory manager" -
w secret -p 389 -h server.example.com -x

dn: cn=managed entries container,ou=containers,dc=example,dc=com
objectclass: top
objectClass: extensibleObject
objectClass: nsContainer
cn: managed entries container
```

2. Add the ***nsslapd-pluginConfigArea*** attribute to the Managed Entries Plug-in entry that points back to the container entry.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 389
-h server.example.com -x

dn: cn=Managed Entries,cn=plugins,cn=config
changetype: modify
add: nsslapd-pluginConfigArea
nsslapd-pluginConfigArea: cn=managed entries container,ou=containers,dc=example,dc=com
```

3. Move or create the definition (Section 6.3.3, "Creating the Managed Entries Instance Definition") and template (Section 6.3.2, "Creating the Managed Entries Template Entry") entries under the new container entry.

6.4. USING VIEWS

Virtual directory tree views, or *views*, create a virtual directory hierarchy, so it is easy to navigate entries, without having to make sure those entries physically exist in any particular place. The view uses information about the entries to place them in the view hierarchy, similarly to members of a filtered role or a dynamic group. Views superimpose a DIT hierarchy over a set of entries, and to client applications, views appear as ordinary container hierarchies.

6.4.1. About Views

Views create a directory tree similar to the regular hierarchy, such as using organizational unit entries for subtrees, but views entries have an additional object class (**nsview**) and a filter attribute (***nsviewfilter***) that set up a filter for the entries which belong in that view. Once the view container entry is added, all of the entries that match the view filter instantly populate the view. The target entries only *appear* to exist in the view; their true location never changes. For example, a view may be created as **ou=Location Views**, and a filter is set for **l=Mountain View**. Every entry, such as **cn=Jane Smith,l=Mountain View,ou=People,dc=example,dc=com**, is immediately listed under the **ou=Location Views** entry, but the real **cn=Jane Smith** entry remains in the **ou=People,dc=example,dc=com** subtree.

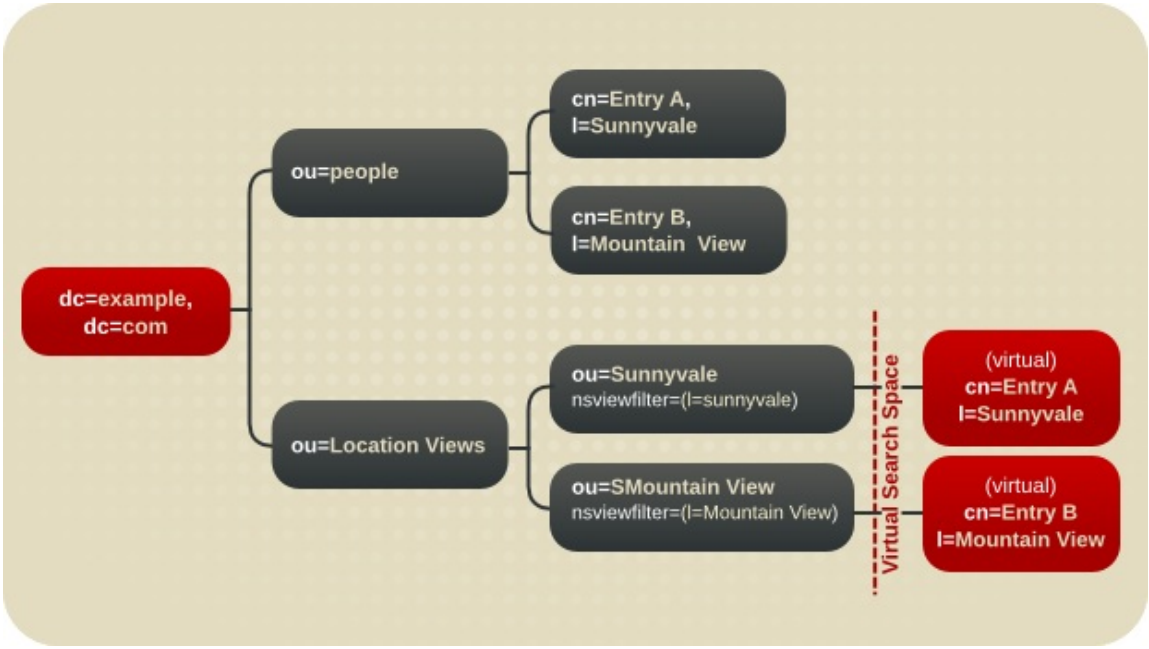
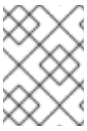


Figure 6.4. A Directory Tree with a Virtual DIT View hierarchy

Virtual DIT views behave like normal DITs in that a subtree or a one-level search can be performed with the expected results being returned.



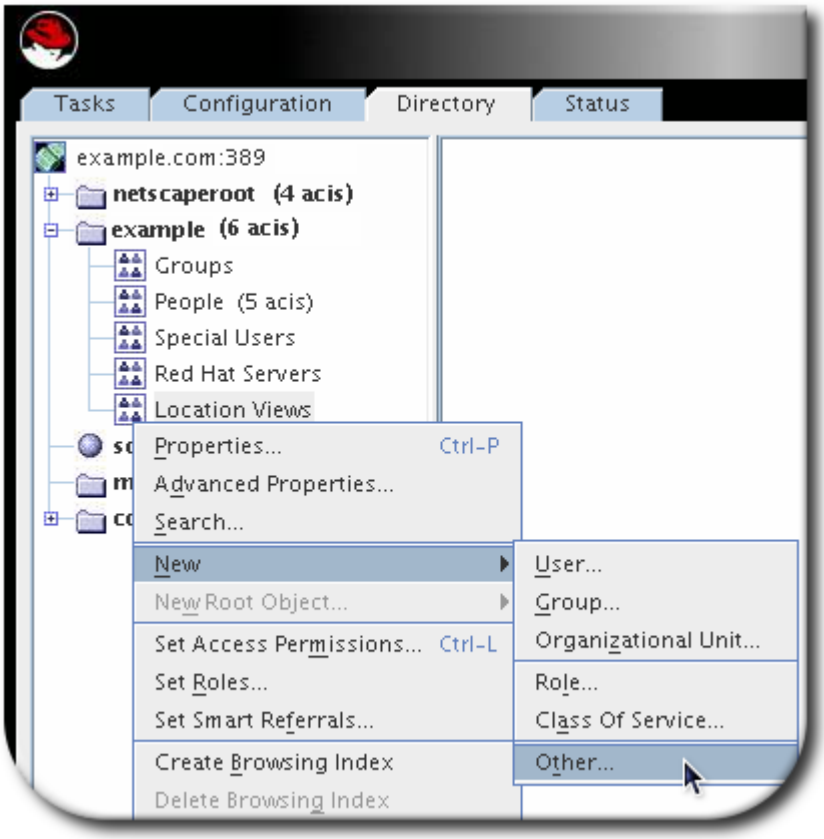
NOTE

There is a sample LDIF file with example views entries, **Example-views.ldif**, installed with Directory Server. This file is in the **/usr/share/dirsrv/data** directory on Red Hat Enterprise Linux 6 (64-bit).

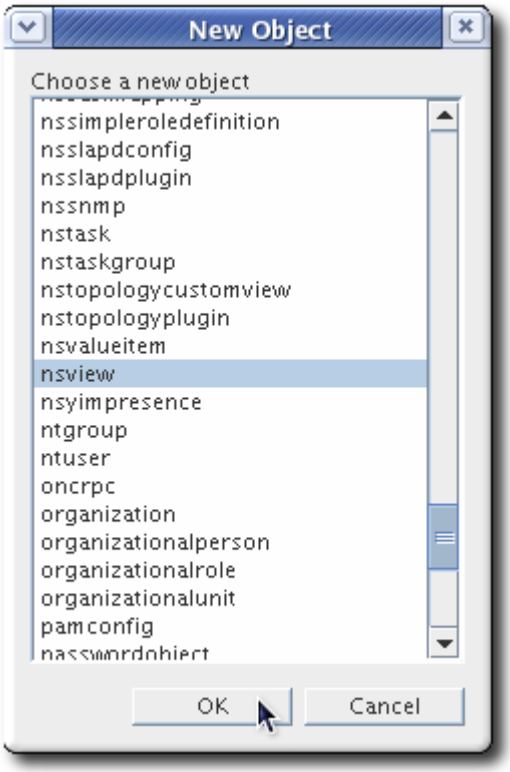
The *Directory Server Deployment Guide* has more information on how to integrate views with the directory tree hierarchy.

6.4.2. Creating Views in the Console

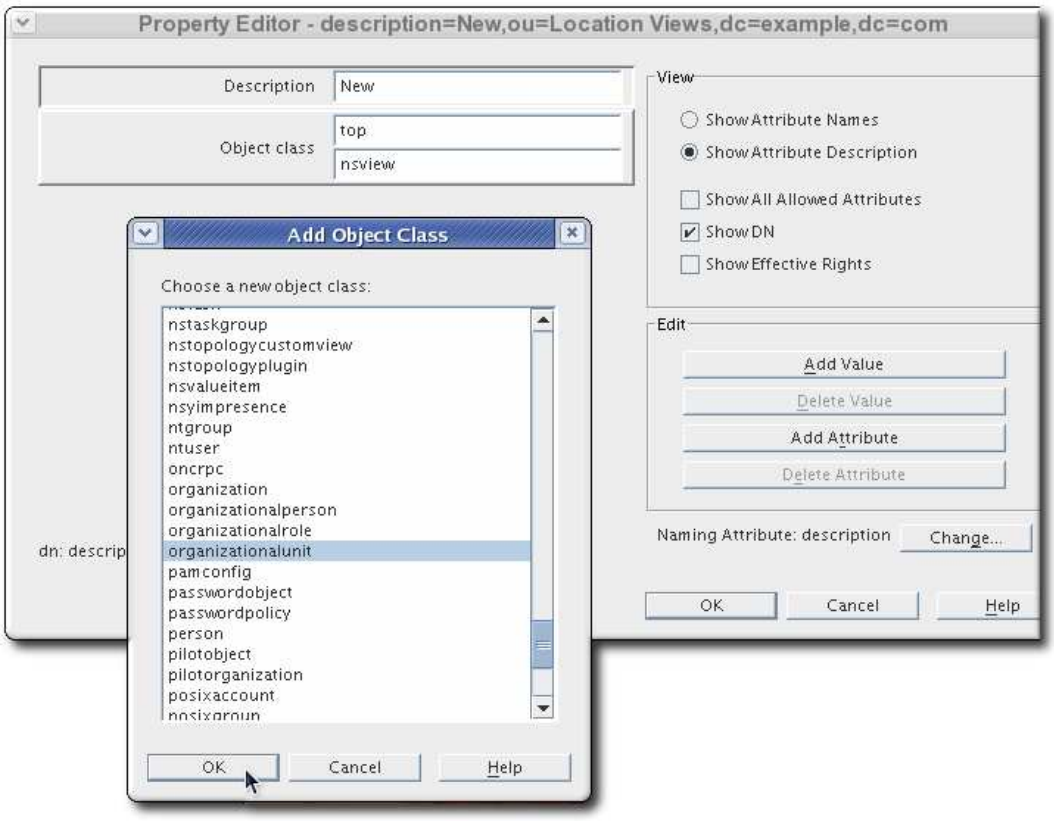
1. Select the **Directory** tab.
2. In the left navigation tree, create an organizational unit suffix to hold the views. For instance, for views based on the locality (**l**) attribute, name this organizational unit **Location Views**. Creating sub suffixes is described in Section 2.1.1.2, "Creating a New Sub Suffix Using the Console" .
3. Right-click **ou=Location Views**, and select **New > Other**.



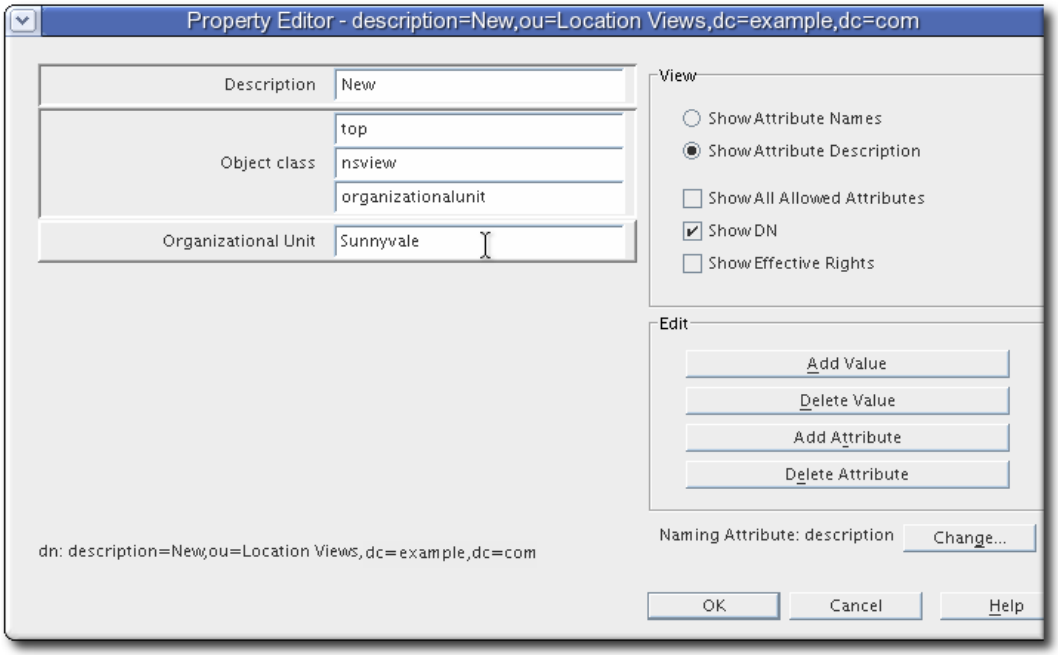
4. Select **nsview** from the **New Object** menu, and click **OK**.



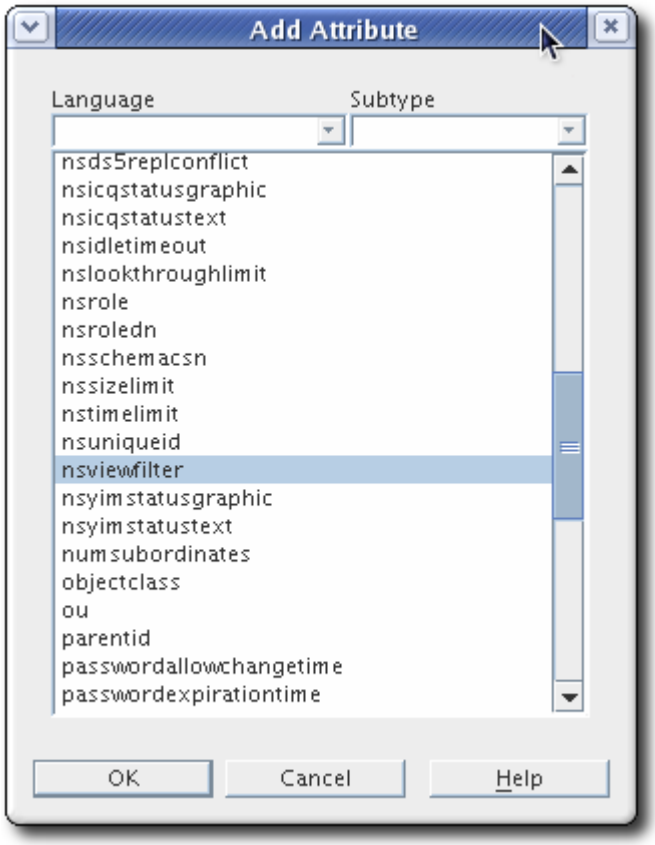
5. In the **Property Editor** window, click the **Add Value** button, and add the organization unit object class.



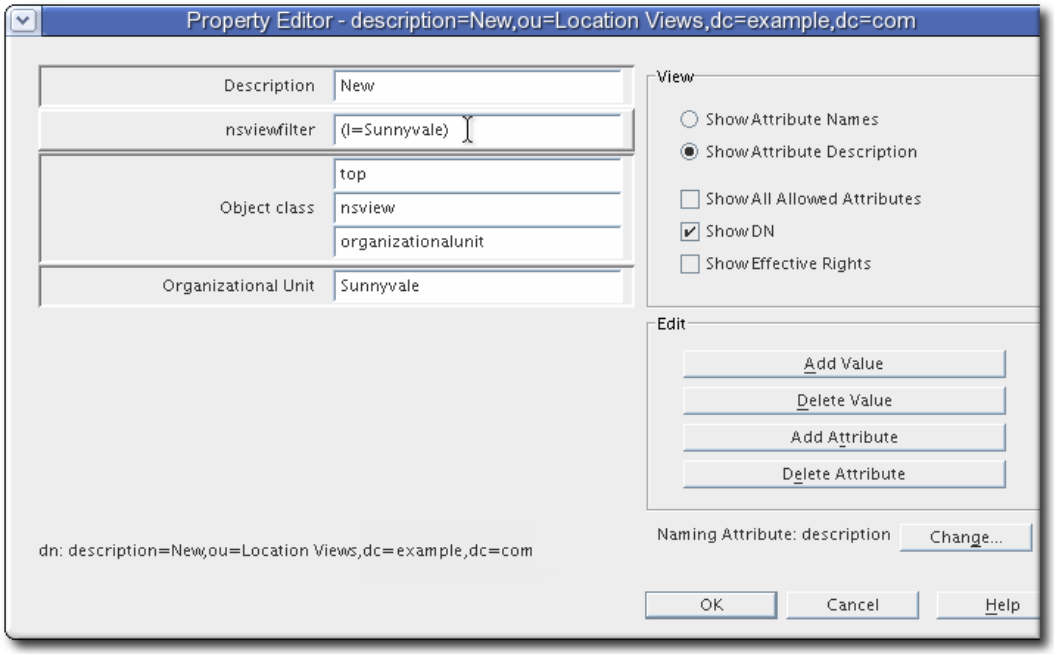
6. Name the organization unit according to how to organize the views. For instance, **ou=Sunnyvale**. Make the ***ou*** attribute the naming attribute.



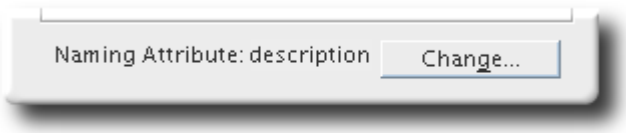
7. Click the **Add Attribute** button, and add the **nsviewfilter** attribute.



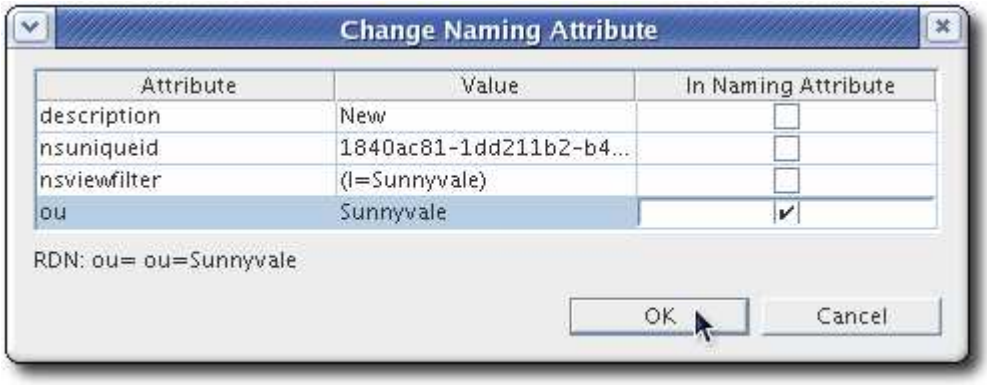
8. Create a filter that reflects the views, such as **(l=Sunnyvale)**.



9. Click the **Change** button in the lower right corner to change the naming attribute.



Use the ***ou*** of the entry as the naming attribute instead of ***description***.



10. Click **OK** to close the attributes box, and click **OK** again to save the new view entry.

The new view is immediately populated with any entries matching the search filter, and any new entries added to directory are automatically included in the view.

6.4.3. Creating Views from the Command Line

1. Use the **ldapmodify** utility to bind to the server and prepare it to add the new view entry to the configuration file.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

2. Add the new views container entry, in this example, under the **dc=example,dc=com** root suffix. This entry must have the **nsview** object class and the ***nsViewFilter*** attribute. The ***nsViewFilter*** attribute sets the attribute-value which identifies entries that belong in the view.

```
dn: ou=Example View,dc=example,dc=com
changetype: add
objectClass: top
objectClass: organizationalUnit
objectClass: nsview
ou: Example View
nsViewFilter: l=Mountain View
description: Example View
```

6.4.4. Improving Views Performance

As Section 6.4.1, "About Views" describes, views are derived from search results based on a given filter. Part of the filter is the attribute defined in the ***nsViewFilter*** attribute; the rest of the filter is based on the entry hierarchy, looking for the ***entryid*** and ***parentid*** of the real entries included in the view.

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

If any of the searched-for attributes – ***entryid***, ***parentid***, or the attribute set in ***nsViewFilter*** – are not indexed, then the views search becomes an unindexed search because the views operation searches the entire tree for matching entries.

To improve views performance, create equality indexes for*entryid*, *parentid*, and the attribute set in *nsViewFilter*.

Creating equality indexes is covered in Section 9.2, "Creating Standard Indexes", and updating existing indexes to include new attributes is covered in Section 9.3, "Applying New Indexes to Existing Databases" .

CHAPTER 7. CONFIGURING SECURE CONNECTIONS

By default, clients and users connect to the Red Hat Directory Server over a standard connection. Standard connections do not use any encryption, so information is sent back and forth between the server and client in the clear.

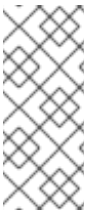
Directory Server supports SSL/TLS connections, Start TLS connection, and SASL authentication, which provide layers of encryption and security that protect directory data from being read even if it is intercepted.

7.1. REQUIRING SECURE CONNECTIONS

The Directory Server has two different ways of encrypting data: SSL/TLS (either through an initial SSL connection or using Start TLS to establish encryption on a previously unencrypted connection) and SASL. If the server is configured to use it, any client can connect to the server using either encryption method to protect data.

For additional security, the Directory Server can be configured to require a certain level of encryption before it allows a connection. The Directory Server can define and require a specific *security strength factor* for any connection. The SSF sets a minimum encryption level, defined by its key strength, for any connection or operation.

To require a minimum SSF for *any* and all directory operations, set the ***nsslapd-minssf*** configuration attribute. When enforcing a minimum SSF, the Directory Server looks at each available encryption type for an operation – SSL/TLS or SASL – and determines which has the higher SSF value and then compares the higher value to the minimum SSF. (It is possible for both SASL authentication and SSL/TLS to be configured for some server-to-server connections, such as replication.)



NOTE

Alternatively, use the ***nsslapd-minssf-exclude-rootdse*** configuration attribute. This sets a minimum SSF setting for all connections to the Directory Server *except* for queries against the root DSE. A client may need to obtain information about the server configuration, like its default naming context, before initiating an operation. The ***nsslapd-minssf-exclude-rootdse*** attribute allows the client to get that information without having to establish a secure connection first.

The SSF for a connection is evaluated when the first operation is initiated on a connection. This allows Start TLS and SASL binds to succeed, even though those two connections initially open a regular connection. After the TLS or SASL session is opened, then the SSF is evaluated. Any connection which does not meet the SSF requirements is closed with an LDAP unwilling to perform error.

Setting a minimum SSF can be used to effectively disable all standard or insecure connections to a directory.

The default ***nsslapd-minssf*** attribute value is 0, which means there is no minimum SSF for server connections. The value can be set to any reasonable positive integer. The value represents the required key strength for any secure connection.

1. Add the ***nsslapd-minssf*** attribute to the **cn=config** entry:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-minssf
nsslapd-minssf: 128
```

2. Restart the server.

```
service dirsrv restart
```

NOTE

An ACI can be set to require an SSF for a specific type of operation, as in Section 13.9.10, “Setting an ACI to Require a Certain Security Strength Factor for Some Operations”.

Secure connections can be required for bind operations by turning on the ***nsslapd-require-secure-binds*** attribute, as in Section 14.8.1, “Requiring Secure Binds”.

7.2. DISABLING SSL AND REQUIRING TLS

TLS is enabled in the ***nsTLS1*** parameter, which is enabled by default.

By default, SSLv3[2] is also enabled. If both TLS and SSLv3 are enabled, then a client can use either protocol to connect to the Directory Server. In some environments, SSLv3 should be disabled so that only TLSv1 connections are allowed.

In the Directory Server configuration, SSLv3 is enabled in the ***nsSSL3*** parameter. There are two ways to change the value of this setting and disable SSLv3:

- Change the ***nsSSL3*** attribute to off. The SSLv3 attribute must be explicitly set to off; if the parameter is missing or has any other value, SSLv3 is implicitly enabled.

  For example:

  ```
  [root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 636
  -h server.example.com -x

  dn: cn=encryption,cn=config
  changetype: modify
  replace: nsSSL3
  nsSSL3: off
  ```

- Use **modutil** to enable FIPS mode, which automatically disables SSLv3. If FIPS mode is enabled, it overrides whatever the ***nsSSL3*** attribute is in order to disable SSLv3. For example:

  ```
  modutil -dbdir /etc/dirsrv/slapd-instance_name -chkfips true
  FIPS mode enabled.
  ```

  Enabling FIPS mode may have other security and configuration implications, however, so it may be easier to disable the SSLv3 parameter.

7.3. MANAGING CERTIFICATES USED BY THE DIRECTORY SERVER

7.3.1. Obtaining and Installing Server Certificates

Before the Directory Server can be set to run in TLS/SSL, server and CA certificates must be properly configured in the Directory Server. If a server certificate has already been generated for the Directory Server instance and the issuing certificate authority (CA) is already trusted by the Directory Server, begin setting up TLS/SSL as described in Section 7.4, “Setting up TLS/SSL”.

Obtaining and installing certificates consists of the following steps:

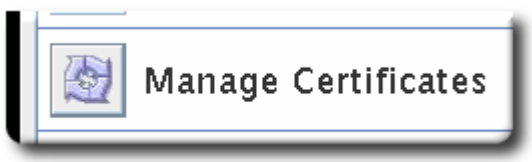
1. Generate a certificate request.
2. Send the certificate request to a certificate authority.
3. Install the server certificate.
4. Set the Directory Server to trust the certificate authority.
5. Confirm that the certificates are installed.

Two wizards automate the process of creating a certificate database and of installing the key-pair. The **Certificate Request Wizard** in the Directory Server Console can generate a certificate request and send it to a certificate authority. The **Certificate Install Wizard** in the Directory Server Console can then install the server certificate and the CA certificate.

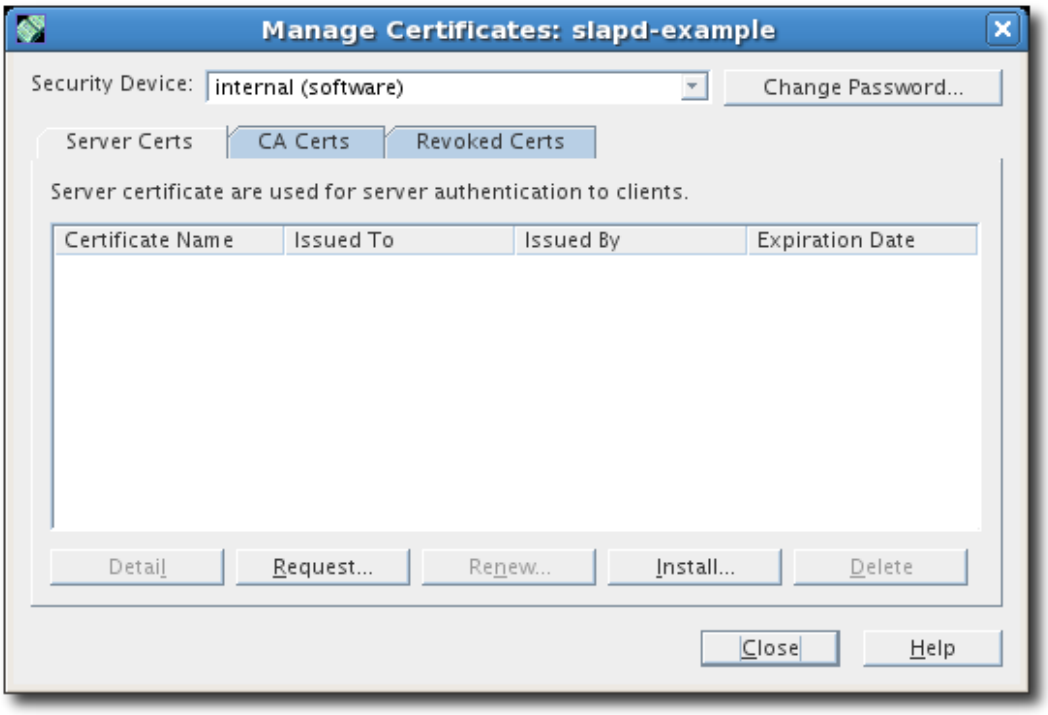
7.3.1.1. Generating a Certificate Request

Generate a certificate request, and send it to a CA. The Directory Server Console has a tool, the **Certificate Request Wizard**, which generates a valid certificate request to submit to any certificate authority (CA).

1. Select the **Tasks** tab, and click **Manage Certificates**.



2. Select the **Server Certs** tab, and click the **Request** button.



3. Enter the **Requester Information** in the blank text fields, then click **Next**.

- *Server Name.* Enter the fully qualified host name of the Directory Server as it is used in DNS and reverse DNS lookups; for example, **dir.example.com**. The server name is critical for client-side validation to work, which prevents man-in-the-middle attacks.
- *Organization.* Enter the legal name of the company or institution. Most CAs require this information to be verified with legal documents such as a copy of a business license.
- *Organizational Unit. Optional.* Enter a descriptive name for the organization within the company.
- *Locality. Optional.* Enter the company's city name.
- *State or Province.* Enter the full name of the company's state or province (no abbreviations).

- *Country.* Select the two-character abbreviation for the country's name (ISO format). The country code for the United States is US.

4. If an external security device is to be used to store the Directory Server certificates, the device is plugged in, and the module has been installed as described in Section 7.8, "Using Hardware Security Modules" , then the module is available in the **Active Encryption Token** menu. The default is to use the software databases with Directory Server, **internal (software)**.

5. Enter the password that will be used to protect the private key, and click **Next**.

The **Next** button is grayed out until a password is supplied.

6. The **Request Submission** dialog box provides two ways to submit a request: directly to the CA (if there is one internally) or manually. To submit the request manually, select **Copy to Clipboard** or **Save to File** to save the certificate request which will be submitted to the CA.

7. Click **Done** to dismiss the **Certificate Request Wizard**.

After generating the certificate request, send it to the CA.

7.3.1.2. Sending a Certificate Request

After the certificate request is generated, send it to a certificate authority (CA); the CA will generate return a server certificate.

1. Most certificate requests are emailed to the CA, so open a new message.

2. Copy the certificate request information from the clipboard or the saved file into the body of the message.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCVXMxEzARBgNVBAgTCkNBTElGT1J
OSUExLDAqBgVBAoTI25ldHNjYXBIlGNvbW11bmljYXRpb25zIGNvcnBvcmF
0aW9uMRwwGgYDVQQDExNtZWxsb24ubmV0c2NhcGUuY29tMIGfMA0GCSqGSI
b3DQEBAQUAA4GNADCBiQKBgQCwAbskGh6SKYOgHy+UCSLnm3ok3X3u83Us7
ug0EfgSLR0f+K41eNqqRftGR83emqPLDOf0ZLTLjVGJaH4Jn4l1gG+JDf/n
/zMyahxtV7+mT8GOFFigFfuxaxMjr2j7IvELlxQ4lfZgWwqCm4qQecv3G+N
9YdbjveMVXW0v4XwIDAQABoAAwDQYK
------END NEW CERTIFICATE REQUEST-----
```

3. Send the email message to the CA.

After emailing the certificate request, wait for the CA to respond with the server certificate. Response time for requests varies. For example, if the CA is internal to the company, it may only take a day or two to respond to the request. If the selected CA is a third-party, it could take several weeks to respond to the request.

After receiving the certificate, install it in the Directory Server's certificate database. When the CA sends a response, be sure to save the information in a text file. The certificate must be available to install in the Directory Server.

Also, keep a backup of the certificate data in a safe location. If the system ever loses the certificate data, the certificate can be reinstalled using the backup file.

7.3.1.3. Installing the Certificate

NOTE

If an existing certificate has the same name as the new certificate you are attempting to install, you cannot use the Directory Server Console to install the certificate. It fails with the error *Internal error: Fail to install certificate -8169.*

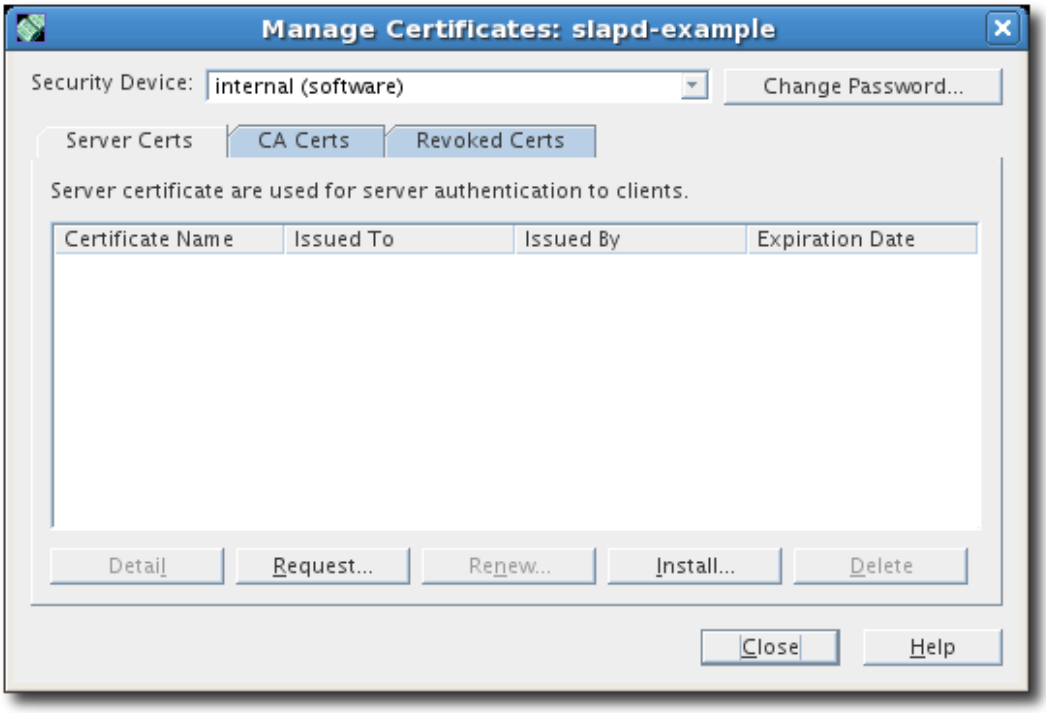
You can use **certutil** to install the certificate.

```
certutil -A -n nickname -t trust-settings -d certDB -f /path/to/certificate_file
```

For more information on using **certutil** to manage certificates, see Section 7.6, “Using certutil”.

1. Select the **Tasks** tab, and click **Manage Certificates**.

2. Select the **Server Certs** tab, and click **Install**.



3. Give the certificate location or paste the certificate text in the text box, then click **Next**.

   - *In this file.* Enter the absolute path to the certificate in this field.
   - *In the following encoded text block.* Copy the text from the CA's email or from the created text file, and paste it in this field.

4. Check that the certificate information displayed is correct, and click **Next**.
5. Give a name to the certificate, and click **Next**.
6. Provide the password that protects the private key. This password is the same as the one provided in step 5 in Section 7.3.1.1, "Generating a Certificate Request" .

After installing the server certificate, configure the Directory Server to trust the CA which issued the server's certificate, Section 7.3.2, "Trusting the Certificate Authority" .

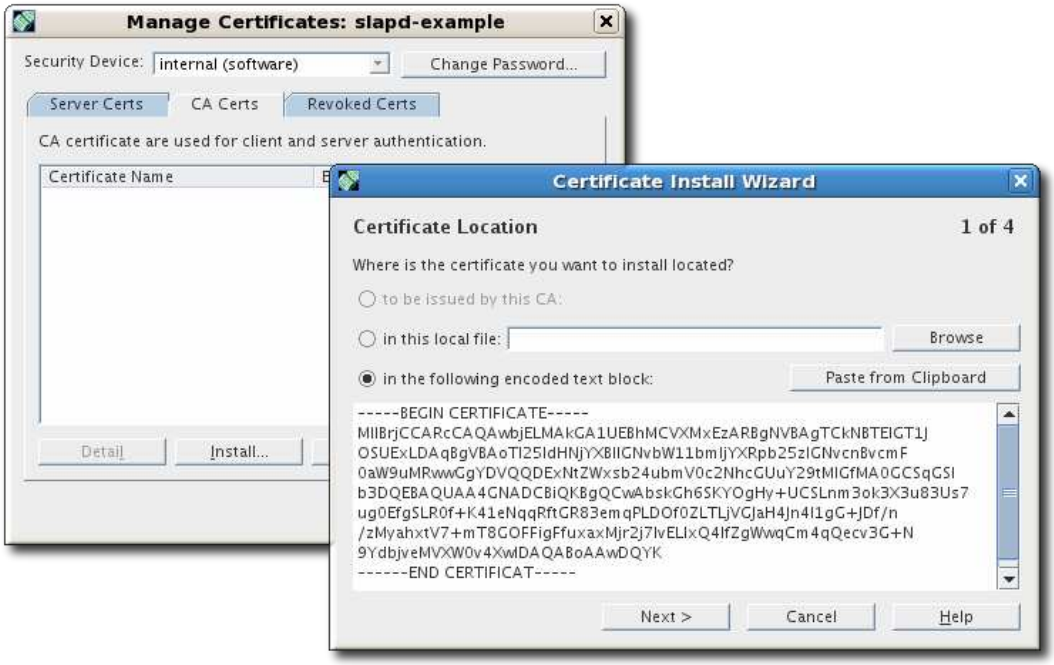
7.3.2. Trusting the Certificate Authority

Configuring the Directory Server to trust the certificate authority consists of obtaining the CA's certificate and installing it into the server's certificate database. This process differs depending on the certificate authority. Some commercial CAs provide a web site that allow users to automatically download the certificate. Others will email it back to users.

After receiving the CA certificate, use the **Certificate Install Wizard** to configure the Directory Server to trust the certificate authority.

1. Select the **Tasks** tab, and click **Manage Certificates**.

2. Go to the **CA Certs** tab, and click **Install**.



3. If the CA's certificate is saved to a file, enter the path in the field provided. Alternatively, copy and paste the certificate, including the headers, into the text box. Click **Next**.

4. Check that the certificate information that opens is correct, and click **Next**.

5. Name the certificate, and click **Next**.

6. Select the purpose of trusting this certificate authority; it is possible to select both options.

- *Accepting connections from clients (Client Authentication).* The server checks that the client's certificate has been issued by a trusted certificate authority.
- *Accepting connections to other servers (Server Authentication).* This server checks that the directory to which it is making a connection (for replication updates, for example) has a certificate that has been issued by a trusted certificate authority.

Once both the server and CA certificates are installed, it is possible to configure the Directory Server to run in TLS/SSL. However, Red Hat recommends verifying that the certificates have been installed correctly.

7.3.3. Renewing Certificates

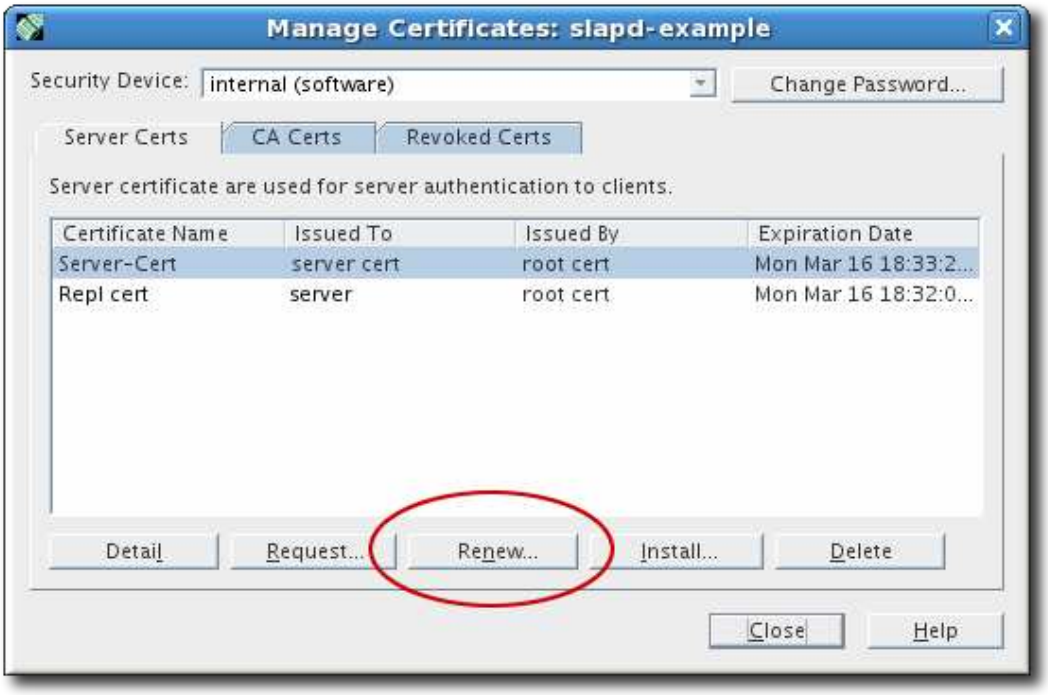
As with any issued identification – drivers' licenses, student IDs – certificates are valid for a predefined period and then expire and must be renewed. To renew a certificate, regenerate a certificate request, using the same information that was used to create the original, submit the request to a CA, and re-install the renewed certificate.

NOTE

When renewing a certificate using the **Certificate Wizard**, the text on the introduction screen does not clearly indicate that the process is renewal and not requesting a new certificate. Also, the requester information is not filled in automatically.

1. Open the Directory Server Console.
2. In the **Tasks** tab, click the **Manage Certificates** button.

3. Click the **Server Certs** tab.
4. Select the certificate to renew from the list of certificates, and click the **Renew** button.

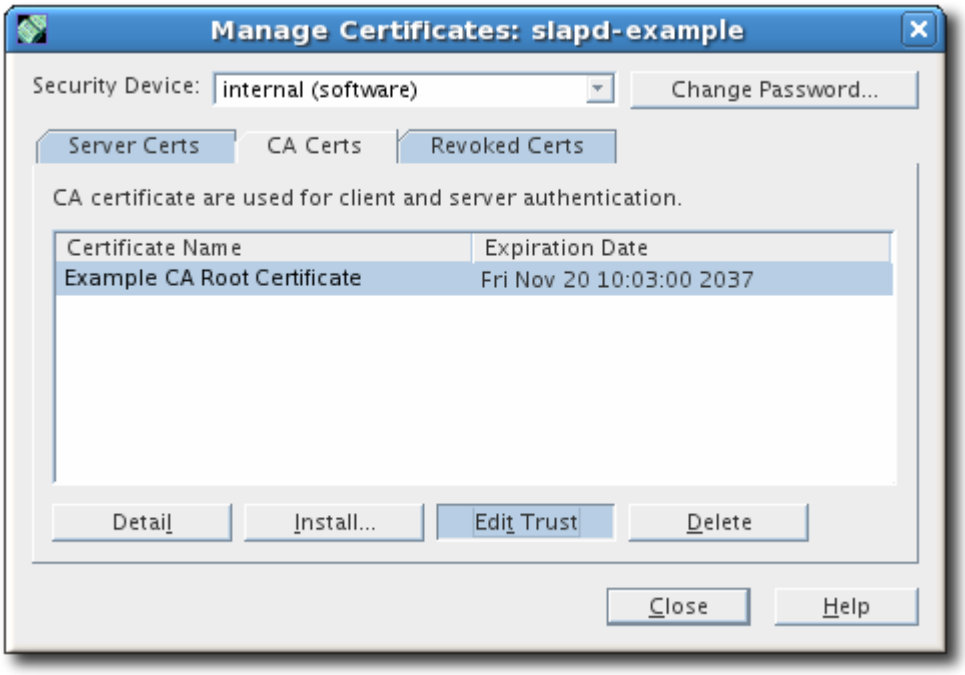


5. Go through the request wizard, using the same information used for requesting the original certificate.
6. Submit the request to a certificate authority.
7. Once the certificate is issued, reinstall it in the Directory Server.
    1. In the **Tasks** tab, click the **Manage Certificates** button.
    2. Click the **Server Certs** tab.
    3. Click the **Install** button.
    4. Paste in the renewed certificate, and continue through the installation wizard.

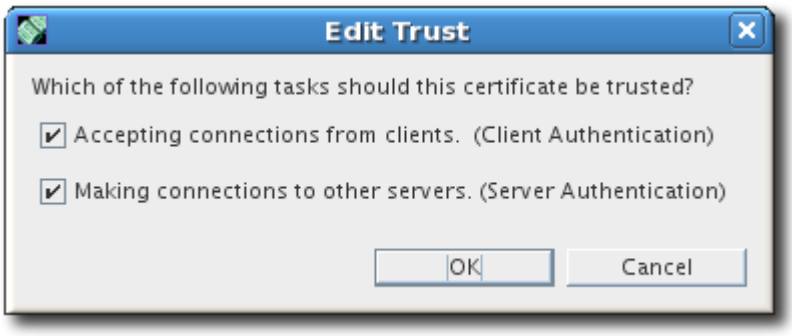
7.3.4. Changing the CA Trust Options

It is sometimes necessary to reject certificates issued by a generally trusted CA. The trust settings on CA certificates installed in the Directory Server can be untrusted, trusted, or change the operations for which it is trusted.

1. In the **Tasks** tab, click the **Manage Certificates** button.
2. Click the **CA Certs** tab.



3. Select the CA certificate to edit.
4. Click the **Edit Trust** button.
5. Set the CA trust options.



- *Accepting connections from clients (Client Authentication).* This option sets whether to accept client, or user, certificates issued by the CA.
- *Making connections to other servers (Server Authentication).* This option sets whether to accept server certificates issued by the CA.
- Click **OK**.

7.3.5. Changing Security Device Passwords

Periodically change the settings for the security databases or devices.

1. In the **Tasks** tab, click the **Manage Certificates** button.

2. In the top of the window, choose a security device from the drop-down list.

3. Click the **Password** button.

4. In the **Change Security Device Password** dialog box, enter the old password, and then enter and confirm the new password.

5. Click **OK**.

7.3.6. Adding Certificate Revocation Lists

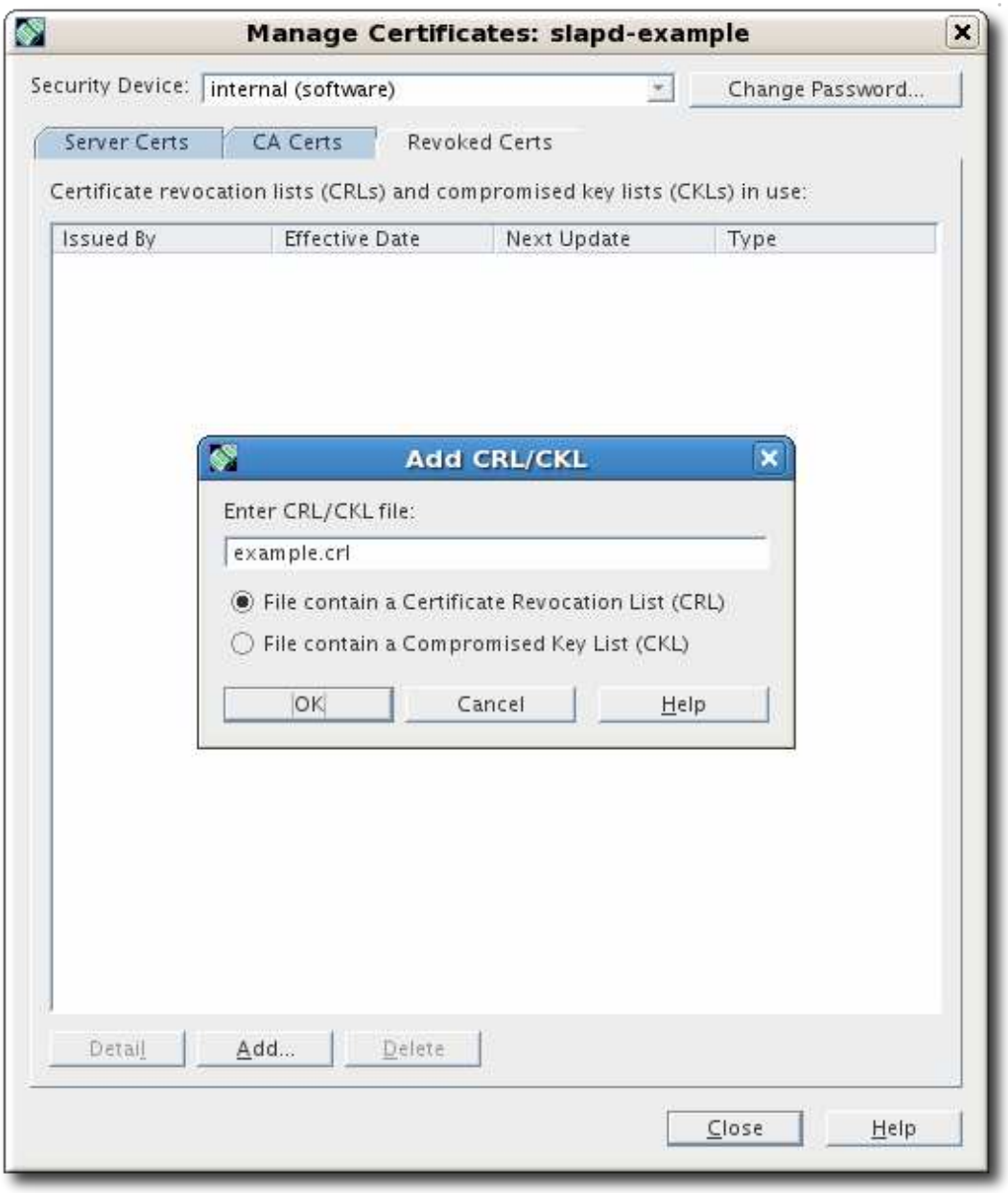
Certificate revocation lists (CRLs) allow CAs to specify certificates that client or server users should no longer trust. If data in a certificate changes, a CA can revoke the certificate and list it in a CRL. CRLs are produced and periodically updated by a CA, so updated CRLs can be added to the Directory Server.

1. Obtain the CRL from the CA; these can usually be downloaded from the CA's website.

2. In the **Tasks** tab, click the **Manage Certificates** button.

3. Select the **Revoked Certs** tab.

4. To add a CRL, click **Add** at the bottom of the window, and enter the full path to the CRL file.

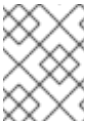


5. Click **OK**.

7.3.7. Managing Certificates Used by the Directory Server Console

The certificates and keys used by the server are stored in NSS security databases in the **/etc/dirsrv/slapd-*instance_name*** directory. The Directory Server Console itself also uses certificates and keys for SSL/TLS connections; these certificates are stored in a separate database in the user's home directory. If the Directory Server Console is used to connect to multiple instances of Directory Server over SSL, then it is necessary to trust every CA which issued the certificates for every Directory Server instance.

When SSL/TLS is enabled for the Directory Server Console (Section 7.4.3, "Enabling TLS/SSL in the Directory Server, Admin Server, and Console"), the Directory Server Console must have a copy of the issuing CA certificate for it to trust the server's SSL client certificates. Otherwise, the Console will return errors about not trusting the CA which issued the certificate.



NOTE

Only the CA certificates for the CA which issued the server's SSL certificate is required. The Directory Server Console does not require its own SSL client certificate.

The Console's security databases are managed directly using **certutil**. The **certutil** tool is described in more detail in Section 7.6, "Using certutil" and in the NSS tool manpages.

To list the certificates in the security database:

```
certutil -d $HOME/.redhat-idm-console -L
```

To add a CA certificate to the database:

```
certutil -d $HOME/.redhat-idm-console -A -t CT,, -a -i /path/to/cacert.asc
```

NOTE

If you are running the Directory Server Console on Windows,

- the security database is located in the **C:\Documents and Settings*user_name*\.389-console** directory.
- change to the **C:\Program Files\Red Hat Identity Management Console** directory to run the **certutil.exe** command.

7.4. SETTING UP TLS/SSL

To provide secure communications over the network, Red Hat Directory Server includes the LDAPS communications protocol. LDAPS is the standard LDAP protocol, running over Transport Layer Security (TLS, formerly Secure Sockets Layer or SSL). Directory Server also allows *spontaneous* secure connections over otherwise-insecure LDAP ports, using the Start TLS LDAP extended operation.

7.4.1. TLS/SSL in Directory Server

The Directory Server supports TLS/SSL to secure communications between LDAP clients and the Directory Server, between Directory Servers that are bound by a replication agreement, or between a database link and a remote database. Directory Server can use TLS/SSL with simple authentication (bind DN and password) or with certificate-based authentication.

Directory Server's cryptographic services are provided by Mozilla Network Security Services (NSS), a library of TLS/SSL and base cryptographic functions. NSS includes a software-based cryptographic token which is FIPS 140-2 certified.

Using TLS/SSL with simple authentication ensures confidentiality and data integrity. There are two major benefits to using a certificate — smart card, token, or software-based — to authenticate to the Directory Server instead of a bind DN and password:

- *Improved efficiency.* When using applications that prompt once for the certificate database password and then use that certificate for all subsequent bind or authentication operations, it is more efficient than continuously providing a bind DN and password.
- *Improved security.* The use of certificate-based authentication is more secure than non-certificate bind operations because certificate-based authentication uses public-key cryptography. Bind credentials cannot be intercepted across the network. If the certificate or device is lost, it is useless without the PIN, so it is immune from third-party interference like phishing attacks.

The Directory Server is capable of simultaneous TLS/SSL and non-SSL communications. This means that you do not have to choose between TLS/SSL or non-SSL communications for the Directory Server; both can be used at the same time. Directory Server can also utilize the Start TLS extended operation to allow TLS/SSL secure communication over a regular (insecure) LDAP port.

There are four steps to enable TLS/SSL:

1. Obtain and install a certificate for the Directory Server, and configure the Directory Server to trust the certification authority's (CA's) certificate.

   For information, see Section 7.3.1, “Obtaining and Installing Server Certificates” .

2. Turn on TLS/SSL in the directory.

   For information, see Section 7.4, “Setting up TLS/SSL”.

3. Configure the Admin Server connect to a TLS-enabled Directory Server.

4. Optionally, ensure that each user of the Directory Server obtains and installs a personal certificate for all clients that will authenticate with TLS/SSL.

   For information, see Section 7.10.1, “Configuring Directory Server to Accept Certificate-Based Authentication from LDAP Clients”.

Most of the time, the server should run with TLS/SSL enabled. If TLS/SSL is temporarily disabled, re-enable it before processing transactions that require confidentiality, authentication, or data integrity.

Before TLS/SSL can be activated, first create a certificate database, obtain and install a server certificate, and trust the CA's certificate, as described in Section 7.3.1, "Obtaining and Installing Server Certificates" .

With TLS/SSL enabled, when the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database. Restarting the Directory Server without the password prompt is possible by using use a hardware crypto device or creating a PIN file (Section 7.4.4, "Creating a Password File for the Directory Server").

NOTE

On TLS-enabled servers, be sure to check the file permissions on certificate database files, key database files, and PIN files to protect the sensitive information they contain. Because the server does not enforce read-only permissions on these files, check the file modes to protect the sensitive information contained in these files.

The files must be owned by the Directory Server user, such as the default **nobody**. If you set a different account during the installation, like Red Hat recommends, use this user and group for a better security instead. The key and cert databases should be owned by the Directory Server user and should typically have read/write access for the owner with no access allowed to any other user (mode **0600**). The PIN file should also be owned by the Directory Server user and set to read-only by this user, with no access to anyone other user (mode **0400**).

WARNING

The Directory Server must already be configured to run in SSL (Section 7.4.2, "Enabling TLS/SSL Only in the Directory Server") and **the server must already have been restarted** before the Directory Server Console can be configured to use SSL. Configuring SSL/TLS for the server requires a server restart to load the new configuration, including the new secure port assignment. However, enabling SSL/TLS for the Console takes effect immediately. Therefore, if the Console has SSL/TLS enabled before the server is running in SSL/TLS, then the Console loses the connection to the server and cannot reconnect.

To disable SSL/TLS in the Directory Server Console, use **ldapmodify** to edit the ***nsServerSecurity*** attribute:

```
nsServerSecurity: off
```

7.4.2. Enabling TLS/SSL Only in the Directory Server

1. Obtain and install CA and server certificates.
2. Set the secure port for the server to use for TLS/SSL communications. In the **Configuration** area, select the **Settings** tab, and enter the value in the **Encrypted Port** field.

The encrypted port number *must not* be the same port number used for normal LDAP communications. By default, the standard port number is **389**, and the secure port is **636**.

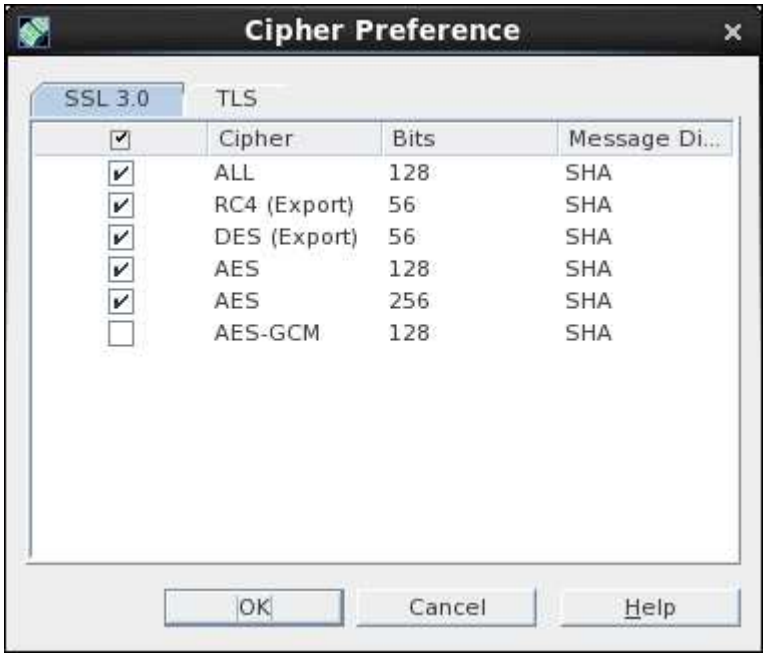
3. Select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane. Select the **Encryption** tab in the right pane.

4. Select the **Enable SSL for this Server** check box.

5. Check the **Use this Cipher Family** check box.

6. Select the certificate to use from the drop-down menu.

7. Click **Cipher Settings**.



By default, **ALL** in the **TLS** tab is selected. When **ALL** is set, the server selects safe ciphers internally.

8. Set the preferences for client authentication.

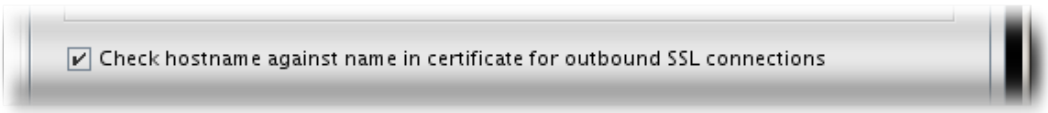
- *Do not allow client authentication.* With this option, the server ignores the client's certificate. This does not mean that the bind will fail.
- *Allow client authentication.* This is the default setting. With this option, authentication is performed on the client's request. For more information about certificate-based authentication, see Section 7.10, "Using Client (Certificate-Based) Authentication".
- *Require client authentication.* With this option, the server requests authentication from the client.

If TLS/SSL is only enabled in the Directory Server and not the Directory Server Console, do not select **Require client authentication** check box.

NOTE

To use certificate-based authentication with replication, the consumer server must be configured either to allow or to require client authentication.

9. To verify the authenticity of requests, select the **Check host name against name in certificate for outbound SSL connections** option. The server does this verification by matching the host name against the value assigned to the common name (**cn**) attribute of the subject name in the being presented for authentication.



By default, this feature is disabled. If it is enabled and if the host name does not match the **cn** attribute of the certificate, appropriate error and audit messages are logged. For example, in a replicated environment, messages similar to these are logged in the supplier server's log files if it finds that the peer server's host name does not match the name specified in its certificate:

```
[DATE] - SSL alert: ldap_sasl_bind("",LDAP_SASL_EXTERNAL) 81 (Netscape runtime error -12276 -
    Unable to communicate securely with peer: requested domain name does not match the server's
    certificate.)
[DATE] NSMMReplicationPlugin - agmt="cn=to ultra60 client auth" (ultra60:1924): Replication
    bind with SSL client authentication failed: LDAP error 81 (Cannot contact LDAP server)
```

Red Hat recommends enabling this option to protect Directory Server's outbound SSL connections against a man-in-the-middle (MITM) attack.

10. Click **Save**.

11. Restart the Directory Server. The Directory Server must be restarted from the command line.

```
service dirsrv restart instance
```

When the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database.

To restart the Directory Server without the password prompt, create a PIN file or use a hardware crypto device. See Section 7.4.4, "Creating a Password File for the Directory Server" for information on how to create a PIN file.

7.4.3. Enabling TLS/SSL in the Directory Server, Admin Server, and Console

1. Obtain server certificates and CA certs, and install them on the Directory Server. This is described in Section 7.3.1, "Obtaining and Installing Server Certificates".

2. Obtain and install server and CA certificates on the Admin Server. This is a similar process as for the Directory Server.

NOTE

It is important that the Admin Server and Directory Server have a CA certificate in common so that they can trust the other's certificates.

3. Configure TLS/SSL for the Directory Server as described in Section 7.4.2, "Enabling TLS/SSL Only in the Directory Server".

4. Require SSL/TLS to connect to the Directory Server Console.

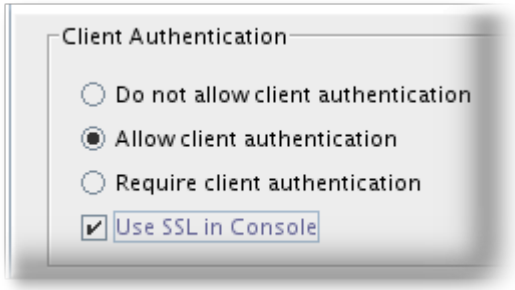
WARNING

The Directory Server must already be configured to run in SSL and **the server must already have been restarted** before the Directory Server Console can be configured to use SSL. Configuring SSL/TLS for the server requires a server restart to load the new configuration, including the new secure port assignment. However, enabling SSL/TLS for the Console takes effect immediately. Therefore, if the Console has SSL/TLS enabled before the server is running in SSL/TLS, then the Console loses the connection to the server and cannot reconnect.

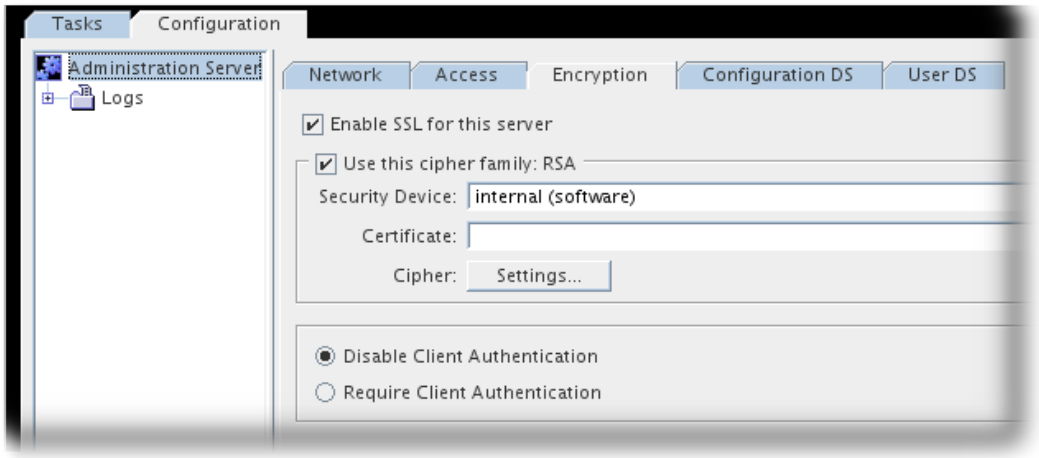
To disable SSL/TLS in the Directory Server Console, use **ldapmodify** to edit the ***nsServerSecurity*** attribute:

```
nsServerSecurity: off
```

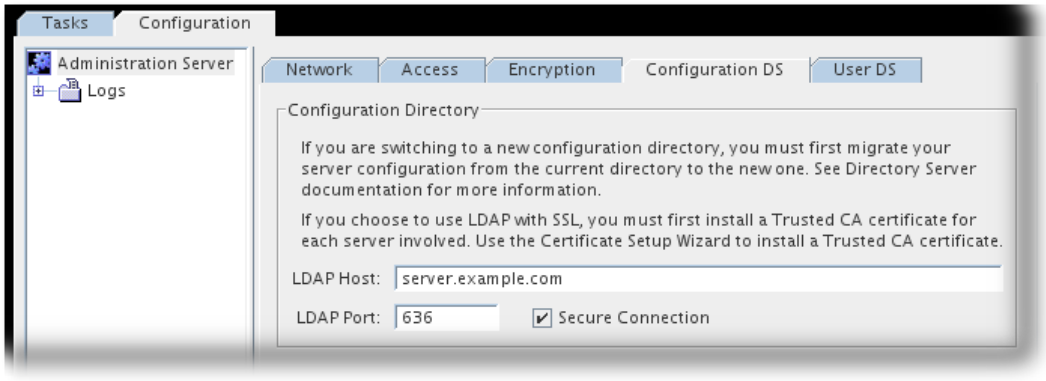
1. Reopen the Directory Server Console.
2. In the **Configuration** tab, select the server, and open the **Encryption** tab.
3. Check the **Use SSL in the Console** box.



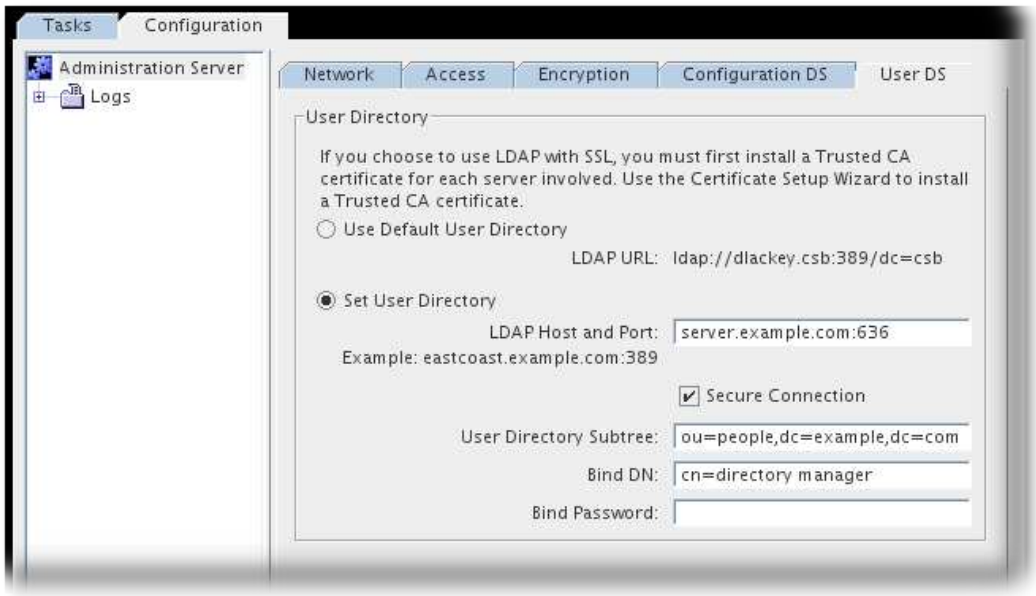
5. In the Admin Server Console, select the **Configuration** tab. Select the **Encryption** tab, check the **Enable SSL** check box, and fill in the appropriate certificate information.



6. In the **Configuration DS** tab, change the port number to the new Directory Server secure port information. See Section 1.6, "Changing Directory Server Port Numbers" for more information. Do this even if the default port of **636** is used. Check the **Secure Connection** check box.



7. In the **User DS** tab, select the **Set User Directory** radio button, and fill in the Directory Server secure port information, the LDAP URL, and the user database information. Check the **Secure Connection** check box.



8. Save the new TLS/SSL settings and **Configuration DS** and **User DS** information in the Admin Server Console.

9. Restart the Admin Server. The server must be restarted from the command line.

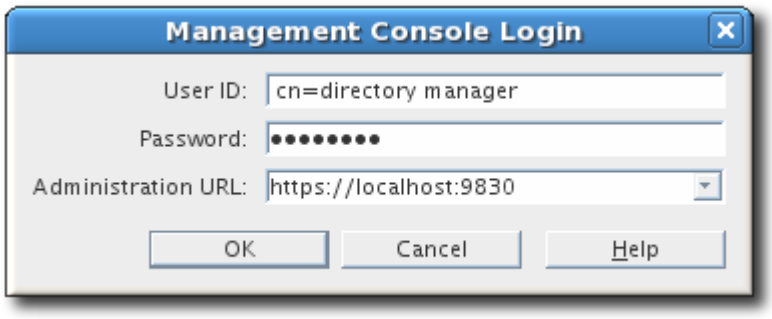
```
service dirsrv-admin restart
```

When the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database.

To restart the Admin Server without the password prompt, create a PIN file or use a hardware crypto device. See Section E.2.9.4, “Creating a Password File for the Admin Server” for information on how to create a PIN file.

NOTE

When next logging into the Directory Server Console, be certain that the address reads **https**; otherwise, the operation will time out, unable to find the server since it is running on a secure connection. After successfully connecting, a dialog box appears to accept the certificate. Click **OK** to accept the certificate (either only for that current session or permanently).



7.4.4. Creating a Password File for the Directory Server

It is possible to store the certificate password in a password file. By placing the certificate database password in a file, the server can be started from the Directory Server Console and also restarted automatically when running unattended.

WARNING

This password is stored in clear text within the password file, so its usage represents a significant security risk. Do not use a password file if the server is running in an unsecured environment.

The password file must be in the same directory where the other key and certificate databases for Directory Server are stored. This is usually the main configuration directory, **/etc/dirsrv/slapd-*instance_name***. The file should be named **pin.txt**.

Include the token name and password in the file. For example:

```
Internal (Software) Token:secret
```

For the NSS software crypto module (the default software database), the token is always called **Internal (Software) Token**.

The PIN file should be owned by the Directory Server user and set to read-only by the Directory Server user, with no access to anyone other user (mode **0400**).

7.4.5. Starting the Directory Server with Expired Certificates

If the Directory Server is configured to run in SSL and its certificate expires, then the Directory Server cannot be started. Because this can cut off access to user and authentication information, as well as other directory data, Directory Server has an option to set how it handles an expired certificate.

The ***nsslapd-validate-cert*** parameter sets how the Directory Server should respond when it attempts to start with an expired certificate:

- **warn** allows the Directory Server to start successfully with an expired certificate, but it sends a warning message that the certificate has expired. This is the default setting.
- **on** validates the certificate and will prevent the server from restarting if the certificate is expired. This sets a hard failure for expired certificates.
- **off** disables all certificate expiration validation, so the server can start with an expired certificate without logging a warning.

To change the behavior of the expired certificate setting, edit the ***nsslapd-validate-cert*** parameter under **cn=config**:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 636 -h
server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-validate-cert
nsslapd-validate-cert: on
```

7.5. COMMAND-LINE FUNCTIONS FOR START TLS

LDAP client tools such as **ldapmodify**, **ldapsearch**, and **ldapdelete** can use TLS/SSL when communicating with an SSL-enabled server or to use certificate authentication. Command-line options also specify or enforce Start TLS, which which allows a secure connection to be enabled on a clear text port after a session has been initiated.

IMPORTANT

For Start TLS to work, the environment variables for the SSL/TLS databases must be configured. This is described in Section A.2, “Using SSL/TLS and Start TLS with LDAP Client Tools” .

In the following example, a network administrator enforces Start TLS for a search for Mike Connor's identification number:

```
ldapsearch -ZZ -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b
"uid=mconnors,ou=people,dc=example,dc=com" "(attribute=govIdNumber)"
```

-ZZ enforces Start TLS.

NOTE

The **-ZZ** option enforces the use of Start TLS, and the server must respond that a Start TLS command was successful. If the **-ZZ** command is used and the server does not support Start TLS, the operation is aborted immediately.

With the **-Z** option, the following errors could occur:

- If there is no certificate database, the operation fails. See Section 7.3.1, “Obtaining and Installing Server Certificates” for information on using certificates.
- If the server does not support Start TLS, the connection proceeds in clear text. To enforce the use of Start TLS, use the **-ZZ** command option.
- If the certificate database does not have the certificate authority (CA) certificate, the connection proceeds in clear text. See Section 7.3.1, “Obtaining and Installing Server Certificates” for information on using certificates.

With the **-ZZ** option, the following errors could occur, causing the Start TLS operation to fail:

- If there is no certificate database. See Section 7.3.1, “Obtaining and Installing Server Certificates” for information on using certificates.
- If the certificate database does not have the certificate authority (CA) certificate. See Section 7.3.1, “Obtaining and Installing Server Certificates” for information on using certificates.
- The server does not support Start TLS as an extended operation.

For SDK libraries used in client programs, if a session is already in TLS mode and Start TLS is requested, then the connection continues to be in secure mode but prints the error **"DSA is unwilling to perform"**.

7.6. USING CERTUTIL

The Directory Server has a command-line tool, **certutil**, which locally creates self-signed CA and client certificates, certificate databases, and keys. The default location for the Directory Server certutil tool is /**usr**/**bin**.

certutil can also be downloaded from https://ftp.mozilla.org/pub/security/nss/releases/.

7.6.1. Creating Directory Server Certificates through the Command Line

The following steps outline how to make the databases, key, CA certificate, server/client certificate, and convert the certificates into **pkcs12** format.

1. Open the directory where the Directory Server certificate databases are stored.

```
[root@server ~]# cd /etc/dirsrv/slapd-instance_name
```

2. Make a backup copy of all of the filed in the directory as a precaution. If something goes awry with while managing certificates, the databases can then be restored. For example:

```
[root@server ~]# tar -cf /tmp/db-backup.tar *
```

3. Create a password file for the security token password.

```
[root@server ~]# vi /tmp/pwdfile

secretpw
```

This password locks the server's private key in the key database and is used when the keys and certificates are first created. The password in this file is also the default password to encrypt PK12 files used by **pk12util**. Because this password is stored in plaintext, the password file should be owned by the user as which Directory Server runs, by default **nobody**. If a different account was set during the installation, like Red Hat recommends, use this account and group instead for a higher security. The password file must be set as read-only for the Directory Server user and allow no access to anyone else (mode **0400**). It's a good idea to have a secure backup of this file.

4. Set the environment variable for the shell to include the **certutil** directory path. For example:

```
[root@server ~]# export PATH=/usr/bin/:$PATH
```

The command varies depending on the shell.

5. Create the key and certificate databases databases.

```
[root@server ~]# certutil -N -d . -f /tmp/pwdfile
```

6. Generate the self-signed CA certificate. **certutil** creates the required key pairs and the certificate. This certificate is used to generate the other server certificates and can be exported for use with other servers and clients.

```
[root@server ~]# certutil -S -n "CA certificate" -s "cn=My Org CA cert,dc=example,dc=com" -2 -x -t "CT,," -m 1000 -v 120 -d . -k rsa -f /tmp/pwdfile
```

7. Generate the Directory Server client certificate.

```
[root@server ~]# certutil -S -n "Server-Cert" -s "cn=FQDN" -c "CA certificate" -t "u,u,u" -m 1001 -v 120 -d . -k rsa -f /tmp/pwdfile
```

The value of the **-s** argument is very important. The leftmost RDN must be **cn=***FQDN* (where *FQDN* is the fully-qualified host and domain name of the Directory Server). For example, to issue a certificate for a server with the name **ldap.example.com**, specify at least **-s "cn=ldap.example.com"**; it is beneficial to have a more descriptive name to help with server identification, such as **"cn=ldap.example.com,ou=DS1"**. The FQDN must be available for DNS and reverse DNS lookups to Directory Server clients because certificate validation may fail if the clients cannot properly resolve the FQDN, and some clients refuse to connect if a server certificate does not have its FQDN in the subject. Additionally, using the format **cn=***hostname.domain* is essential for Directory Server clients to protect themselves from man in the middle attacks.

NOTE

There should only be one ***cn*** in a certificate subject name. To add detail to the subject name, use ***cn*** as the RDN and another attribute – like ***ou***, ***l***, or ***c*** – for the other subject name elements.

To provide a subjectAltName, as well as the nickname, use the **-8** argument in addition to the **-s** argument.

To use the Directory Server behind a DNS round robin or any other scheme which aliases a single server certificate to multiple host names, see the TLS/SSL information about server name wildcards or subjectAltName.

Server certificates for other servers are created using a similar command as for the Directory Server certificate. Make sure that every **-n** option (nickname) and **-m** option (serial number) is unique for every certificate, and make sure that the **-s** option gives the correct FQDN for the server.

NOTE

Keep careful track on the numbers set with the **-m** option. The **-m** option sets the unique identifier for the server certificate, and a CA cannot issue two certificates with the same ID. Keep a log of issued serial numbers so that no number is ever duplicated.

8. Export the CA certificate for use with other servers and clients. A client usually requires the CA certificate to validate the server certificate in an TLS/SSL connection. Use **certutil** to export the CA certificate in ASCII/PEM format:

```
[root@server ~]# certutil -d . -L -n "CA certificate" -a > cacert.asc
```

The way that the CA certificate is imported is different for every client. For example, **certutil** can import a CA certificate into another Directory Server certificate database:

```
[root@server ~]# cd /etc/dirsrv/slapd-otherserver
[root@server ~]# certutil -A -d . -n "CA certificate" -t "CT,," -a -i cacert.asc
```

9. Use **pk12util** to export other server certificates and keys created with **certutil** so that they can be used on a remote server.

```
[root@server ~]# pk12util -d . -o ldap1.p12 -n Server-Cert -w /tmp/pwdfile -k /tmp/pwdfile
```

The **-w** argument is the password used to encrypt the **.p12** file for transport. The **-k** argument specifies the password for the key database containing the server certificate being exported to **.p12**.

10. If the Directory Server will run with TLS/SSL enabled, then create a password file (**pin.txt**) for the server to use so it will not prompt you for a password every time it restarts. Creating the password file is described in Section 7.4.4, “Creating a Password File for the Directory Server” .

The certificates created by **certutil** are automatically available in the **Encryption** tab of the Console. There is no need to import them because they are already in the certificate database.

7.6.2. certutil Usage

certutil can be used for a variety of tasks to manage certificates and keys, such as generating certificate requests and removing certificates from the certificate database. Some of the most common options are listed in Table 7.1, “certutil Options”. For the full list of commands and arguments, run **certutil -H** from the command line.

Table 7.1. certutil Options

Options	Description
-x	Creates a self-signed CA certificate.
-S	Creates a server or client certificate.
-R	Generates a certificate request.
-N	Creates new security databases.
-L	Lists all of the certificates in the database.
-A	Adds a certificate to the certificate database.
-n	Gives the name of the certificate.

Options	Description
-d	Certificate database directory; this is the directory for the subsystem instance.
-m	The serial number for the certificate.
-k	The key type to use; the only option is **rsa**.
-g	The key size. The recommended size for RSA keys is 2048.
-s	The subject name of the certificate.
-t	The trust arguments for the certificate, meaning the purposes for which the certificate is allowed to be used.
-v	The validity period, in months.
numbers 1-8	These set the available certificate extensions. Only eight can be specified through the **certutil** tool: • Key Usage: 1 • Basic Constraints: 2 • Certificate Authority Key ID: 3 • CRL Distribution Point: 4 • Netscape Certificate Type: 5 • Extended Key Usage: 6 • Email Subject Alternative Name: 7 • DNS Subject Alternative Name: 8
-a	Outputs the certificate request to an ASCII file instead of binary.
-o	The output file to which to save the certificate request.
-i	An input file containing a certificate.
-f	The path to a password file for the security databases password.

Table 7.2, "certutil Examples" has some of the common uses for the **certutil** command.

Table 7.2. certutil Examples

Example	Description
certutil -L -d .	Lists the certificates in the database.
certutil -N -d .	Creates new key (**key3.db**) and certificate (**cert8.db**) databases.
certutil -S -n "CA certificate" -s "cn=My Org CA cert,dc=example,dc=com" -2 -x -t "CT,," -m 1000 -v 120 -d . -k rsa	Creates a self-signed CA certificate.
certutil -S -n "Server-Cert" -s "cn=*FQDN*" -c "CA certificate" -t "u,u,u" -m 1001 -v 120 -d . -k rsa	Creates a client certificate.

Example	Description
certutil -L -d . -n "*cert_name*"	"Pretty prints" the specified certificate; the *cert_name* can specify either a CA certificate or a client certificate.
certutil -L -d . -n "*cert_name*" > certfile.asc	Exports the specified certificate out of the database to ASCII (PEM) format.
certutil -L -d . -n "*cert_name*" -r > certfile.bin	Exports the specified certificate out of the database to binary format; this can be used with Directory Server attributes such as ***usercertificate;binary***.

7.7. UPDATING ATTRIBUTE ENCRYPTION FOR NEW SSL/TLS CERTIFICATES

When TLS/SSL is first configured, there is no problem with attribute encryption. However, if the TLS/SSL certificate is changed, then attribute encryption fails, with messages like these:

```
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - attrcrypt_unwrap_key: failed to unwrap key for
cipher AES
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to retrieve key for cipher AES in
attrcrypt_cipher_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to initialize cipher AES in attrcrypt_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - attrcrypt_unwrap_key: failed to unwrap key for
cipher AES
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to retrieve key for cipher AES in
attrcrypt_cipher_init
Apr  4 13:00:35 smtp1 logger: [04/Apr/2017:13:00:34 -0700] - Failed to initialize cipher AES in attrcrypt_init
```

This is because the previously-generated keys do not work with the new server certificate. To correct these errors, force the server to generate new keys for attribute encryption:

1. Stop the server.

   ```
   service dirsrv stop
   ```

2. Open the **dse.ldif** file.

   ```
   vim /etc/dirsrv/dse.ldif
   ```

3. There are special encryption key entries for the encryption ciphers used for attribute encryption under the database configuration. For example:

   ```
   dn: cn=AES,cn=encrypted attribute keys,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
   objectClass: top
   objectClass: extensibleObject
   cn: AES
   nssymmetrickey::
   mSLm/RlCLvPZrSdARHPowedF9zKx+kjVTww5ARE4w0lbl2YlYvrIg6mUlSsmzMfdhc1BBURhFDNwwUDisH
   WwiMJRlvHXstx5EGstWE9xokIU+xeMZF8cPJrY1udFSPFc0iKyiCOaPacWpomn/XPaVxkQqBWk4vJzrHHhH
   1o3bNg=
   ```

   Delete these entries.

4. Start the server again.

   ```
   service dirsrv start
   ```

   As soon as the server restarts, it generates new encryption keys for the encrypted attribute keys.

7.8. USING HARDWARE SECURITY MODULES

A security module serves as a medium between the Directory Server and the SSL layer. The module stores the keys and certificates used for encryption and decryption. The standard which defines these modules is Public Key Cryptography Standard (PKCS) #11, so these modules are PKCS#11 modules.

By default, Directory Server uses built-in security databases, **key3.db** and **cert8.db**, to store the keys and certificates used by the servers.

It is also possible to use external security devices to store Directory Server certificates and keys. For Directory Server to use an external PKCS#11 module, the module's drivers must be installed in Directory Server.

NOTE

It is possible to install PKCS #11 modules through the Directory Server Console, but it is recommended that administrators use the command line to add modules. Because of differences in the way manufacturers implement their modules, some modules (such as nCipher external tokens) must be loaded through the command line rather than the Directory Server Console. For all modules, using the command line is an easier, more effective way to add modules.

7.8.1. Installing PKCS#11 Modules Through the Command Line

Installing PKCS#11 modules through the command line requires the NSS **modutil** tool.

1. Connect the external security device, and install its drivers on the server machine.
2. Use the **-add** and **-libfile** arguments with **modutil** to load the PKCS#11 module and it associated libraries. The *module_name* is the name of the module given when the security device's drivers were installed; the *library_file* is the full path to the module's library.

   ```
   [root@server ~]# modutil -dbdir /etc/dirsrv/slapd-instance_name/ -nocertdb -add module_name -libfile
   library_file
   ```

   For example, for an nCipher hardware security module:

   ```
   [root@server ~]# modutil -dbdir /etc/dirsrv/slapd-instance_name/ -nocertdb -add nethsm -libfile
   /opt/nfast/toolkits/pkcs11/libcknfast.so
   ```

3. List the loaded modules to make sure that the new PKCS#11 module was added correctly.

   ```
   [root@server ~]# modutil -list -dbdir /etc/dirsrv/slapd-instance_name/

   Listing of PKCS #11 Modules
   -----------------------------------------------------------
    1. NSS Internal PKCS #11 Module
          slots: 2 slots attached
         status: loaded

          slot: NSS Internal Cryptographic Services
         token: NSS Generic Crypto Services

          slot: NSS User Private Key and Certificate Services
         token: NSS Certificate DB

    2. nCipher NetHSM PKCS #11 Module
          slots: 2 slots attached
         status: loaded

          slot: nethsm external cryptographic services
         token: nethsm crypto services

          slot: nethsm user private key and certificate services
         token: nethsm certificate db
   -----------------------------------------------------------
   ```

7.8.2. Adding Certificates to an HSM

All of the certificates required for secure connections must be imported into the HSM. Because of SELinux requirements, the best method to import certificates into the Directory Server databases is using the **certutil** command.

At a minimum, you will need to import the Directory Server and Admin Server server certificates, as well as the CA certificate for the CA which issued those certificates. If there are other certificates used by the Directory Server or Admin Server, then those must also be imported.

For example:

```
certutil -A -d /etc/dirsrv/slapd-instance_name -n "CA certificate" -t "CT,," -a -i /tmp/cacert.asc

certutil -A -d /etc/dirsrv/slapd-instance_name -n "Server-Cert" -t "u,u,u" -a -i /tmp/cert.asc
```

certutil is described more in Section 7.6, “Using certutil”.

7.8.3. Setting SELinux Policies for HSMs

In most situations, it will be necessary to create an SELinux policy module to allow the Directory Server to use the certificates stored in the HSM. However, the requirements for writing those policies are different depending on the vendor and model of the HSM.

For a basic primer in writing SELinux policies, see Dan Walsh's article on writing SELinux modules. The Red Hat Enterprise Linux documentation contains information on troubleshooting policy problems in the SELinux Guide. For additional help with writing HSM-related policies, contact Red Hat support.

7.9. SETTING ENCRYPTION CIPHERS

The Directory Server supported several different ciphers, and the type of ciphers to use for TLS/SSL communications are set by the user. A *cipher* is the algorithm used in encryption. Some ciphers are more secure, or stronger, than others. Generally speaking, the more bits a cipher uses during encryption, the more difficult it is to decrypt the key.

When a client initiates an TLS/SSL connection with a server, the client tells the server what ciphers it prefers to use to encrypt information. In any two-way encryption process, both parties must use the same ciphers. There are a number of ciphers available. The server needs to be able to use the ciphers that will be used by client applications connecting to the server.

7.9.1. Available Ciphers

This section lists information about the available ciphers for Directory Server encryption. Each cipher has the following information:

- *Directory Server name.* The name of the cipher suite used when configuring the Directory Server. The Directory Server uses this name both internally and in the Directory Server Console.
- *Key exchange.* The key exchange algorithm. DHE stands for Diffie-Hellman; DSS stands for Digital Signature Standard. The 1024 bit ciphers are lower strength ciphers formerly used for export control.
- *Encryption Algorithm.* AES stands for the Advanced Encryption Standard. DES stands for Data Encryption Standard.
- *Symmetric Key Bit Size.* The size in bits of the key used for the actual transport data encryption.
- *Message Authentication.* SHA stands for Secure Hash Algorithm.

NOTE

Directory Server supports ciphers for TLSv1 (recommended) and SSLv3. SSLv2 support is deprecated and not enabled by default in Directory Server.

To get a list of ciphers supported by the available version of the crypto library:

```
$ ldapsearch -xLLL -H ldap://localhost:389 -D "cn=Directory Manager" -W \
    -b 'cn=encryption,cn=config' -s base nsSSLSupportedCiphers -o ldif-wrap=no

dn: cn=encryption,cn=config
nsSSLSupportedCiphers: TLS::tls_rsa_aes_256_sha::AES::SHA1::256
nsSSLSupportedCiphers: TLS::rsa_aes_256_sha::AES::SHA1::256
...
```

To get a list of ciphers currently configured in **cn=encryption,cn=config** by the ***nsSSL3Ciphers*** parameter:

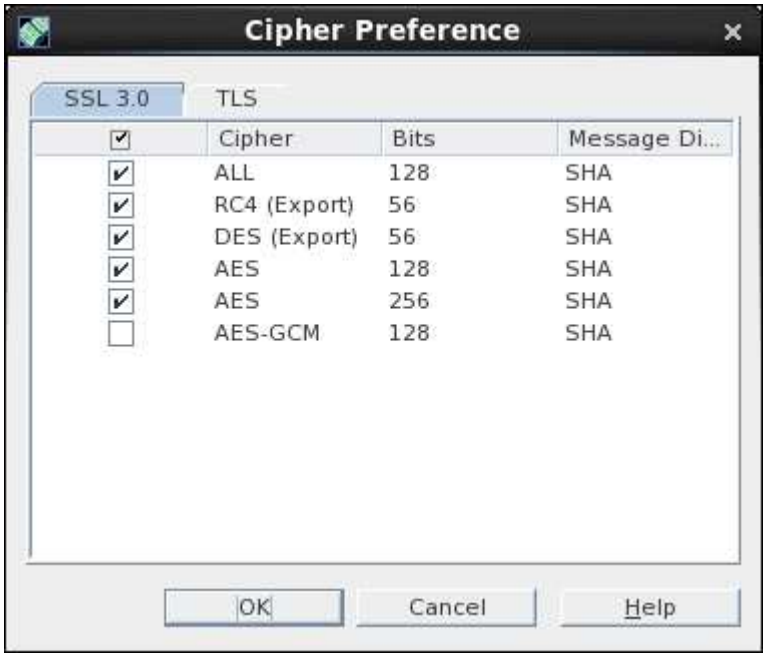
```
$ ldapsearch -xLLL -H ldap://localhost:389 -D "cn=Directory Manager" -W \
    -b 'cn=encryption,cn=config' -s base nsSSLEnabledCiphers -o ldif-wrap=no

dn: cn=encryption,cn=config
```

```
nssslenabledciphers: TLS::tls_dhe_dss_aes_256_sha::AES::SHA1::256
nssslenabledciphers: TLS::tls_dhe_rsa_aes_256_sha::AES::SHA1::256
...
```

7.9.2. Selecting the Encryption Cipher

1. Make sure TLS/SSL is enabled for the server. For instructions on enabling TLS/SSL, see Section 7.4, "Setting up TLS/SSL".
2. Select the **Configuration** tab, and then select the topmost entry in the navigation tree in the left pane.
3. Select the **Encryption** tab in the right pane.
4. Click the **Cipher Setting** button.

5. In the **Cipher Preference** dialog box, specify which ciphers for the Directory Server to use by selecting them from the list, and click **OK**.



By default, **ALL** in the **TLS** tab is selected. When **ALL** is set, the server selects safe ciphers internally.

WARNING

Avoid selecting the **none,MD5** cipher because the server will use this option if no other ciphers are available on the client, instead of refusing the connection. The **none,MD5** cipher is not secure because encryption does not occur.

7.10. USING CLIENT (CERTIFICATE-BASED) AUTHENTICATION

Directory Server allows certificate-based authentication for the command-line tools (which are LDAP clients) and for server-to-server connections (replication and chaining). Three things are required for the Directory Server to allow client authentication:

- The server must have SSL turned on. See Section 7.4, "Setting up TLS/SSL" for more information.
- The Directory Server must trust the CA who issued the certificate to the client, as described in step 6 of Section 7.3.2, "Trusting the Certificate Authority".
- The subject DN in the certificate must be mapped in the user DN through a mapping in the **certmap.conf** file, as in Section 7.10.2, "Mapping DNs to Certificates".

NOTE

A single configuration parameter, ***nsslapd-certdir***, in **cn=config** in **dse.ldif** lists the directory containing the key, certificate, and security files. The directory name should be unique and specific to the server. For example, the **/etc/dirsrv/slapd-*instance_name*** directory contains the key and certificate databases only for the Directory Server instance called *instance_name*. That directory will not contain key and certificate databases for any other server or client, nor will any of the key, certificate, or other security-related files for *instance_name* be located in any other directory.

The certificate and key files have been consolidated into a single file, specified in the ***nsslapd-certdir*** parameter, and the key and certificate file is stored in the **/etc/dirsrv/slapd-*instance_name*** directory.

When a server receives a request from a client, it can ask for the client's certificate before proceeding.

After checking that a client certificate chain ends with a trusted CA, the server can optionally determine which user is identified by the client certificate and then look up that user's entry in the directory. Each certificate has the name of the identity it verifies in a subject name, called the *subject DN*. The server authenticates the user by comparing the information in the subject DN with the DN of the user's directory entry.

In order to locate user entries in the directory, a server must know how to interpret the subject names of certificates from different CAs. The mapping between the subject names of the certificates and the user DNs is defined in the **certmap.conf** file. This file provides three kinds of information for each listed CA:

- It maps the distinguished name (DN) in the certificate to a branch point in the LDAP directory.
- It specifies which DN values from the certificate (user name, email address, and so on) the server should use for the purpose of searching the directory.
- It specifies whether the server should go through an additional verification process. This process involves comparing the certificate presented for authentication with the certificate stored in the user's directory entry. By comparing the certificate, the server determines whether to allow access or whether to revoke a certificate by removing it from the user's entry.

If more than one directory entry contains the information in the user's certificate, the server can examine all matching entries in order to determine which user is trying to authenticate. When examining a directory entry, the server compares the presented certificate with the one stored in the entry. If the presented certificate does not match any certificates or if the matching entries do not contain certificates, client authentication fails.

After the server finds a matching entry and certificate in the directory, it can determine the appropriate kind of authorization for the client. For example, some servers use information from a user's entry to determine group membership, which in turn can be used during evaluation of ACIs to determine what resources the user is authorized to access.

7.10.1. Configuring Directory Server to Accept Certificate-Based Authentication from LDAP Clients

Client authentication to the Directory Server will require or allow a user to use a certificate to establish its identity, in addition to the server having to present a certification. This is also called *certificate-based authentication*.

1. On the client system, obtain a client certificate from the CA.
2. Install the client certificate on the client system.

   Regardless of how the certificate is sent (either in email or on a web page), there should be a link to click to install the certificate.

Record the certificate information that is sent from the CA, especially the subject DN of the certificate because the server must be configured to map it to an entry in the directory. The client certificate resembles the following:

```
-----BEGIN CERTIFICATE-----
MIICMjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBh
MCVVMxIzAhBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0w
GwYDVQQLExRXaWRnZXQgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdC
BUZXN0IFRlc3QgVGVzdCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3
WhcNOTgwMzI2MDIzMzU3WjBPMQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTm
V0c2NhcGUgRGlyZWN0b3
------END CERTIFICATE-----
```

3. Convert the client certificate into binary format using the **certutil** utility.

```
certutil -L -d certdbPath -n userCertName -r > userCert.bin
```

certdbPath is the directory which contains the certificate database; for example, a user certificate for Mozilla Thunderbird is stored in **$HOME/.thunderbird**. *userCertName* is the name of the certificate, and *userCert***.bin** is the name of the output file for binary format.

4. On the server, map the subject DN of the certificate to the appropriate directory entry by editing the **certmap.conf** file.

NOTE

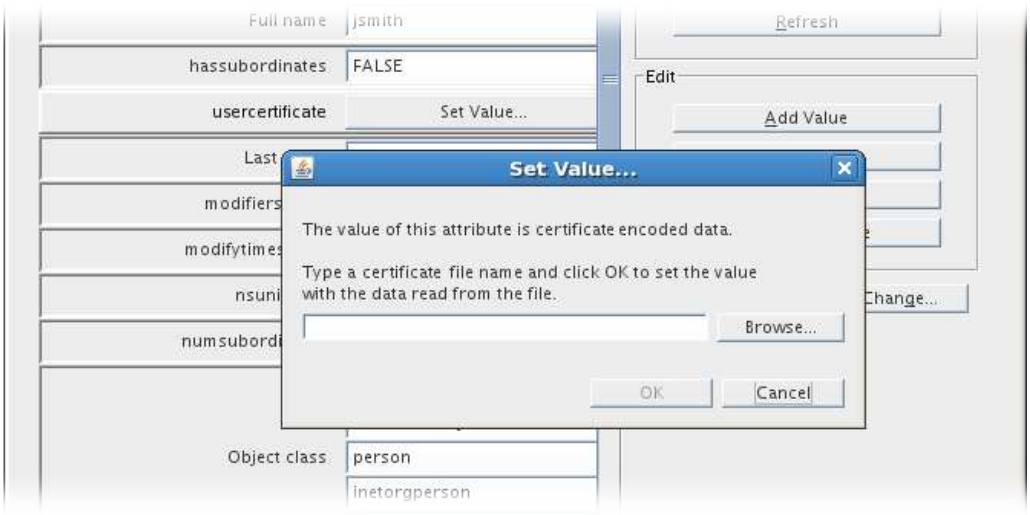
Do not map a certificate-based authentication certificate to a distinguished name under **cn=monitor**. Mapping a certificate to a DN under **cn=monitor** causes the bind operation to fail. Map the certificate to a target located elsewhere in the directory information tree. Make sure that the ***verifyCert*** parameter is set to **on** in the **certmap.conf** file. If this parameter is not set to **on**, Directory Server simply searches for an entry in the directory that matches the information in the **certmap.conf** file. If the search is successful, it grants access without actually checking the value of the ***userCertification*** and ***usercertificate;binary*** attributes.

5. In the Directory Server, modify the directory entry for the user or identity (if it is another server) who owns the client certificate to add the ***usercertificate*** attribute.

    1. Select the **Directory** tab, and navigate to the user entry.

    2. Double-click the user entry, and use the **Property Editor** to add the ***usercertificate*** attribute, with the **binary** subtype.

        When adding this attribute, instead of an editable field, the server provides a **Set Value** button.



    3. Click **Set Value**.

        A file selector opens. Use it to select the binary file created in step 3.

For information on using the Directory Server Console to edit entries, see Section 3.1.3, "Modifying Directory Entries".

For information on how to use TLS/SSL with **ldapmodify**, **ldapdelete**, and **ldapsearch**, see Section A.2, "Using SSL/TLS and Start TLS with LDAP Client Tools".

7.10.2. Mapping DNs to Certificates

When a server performs client authentication, it interprets a certificate, extracts user information, and then searches the directory for that information. In order to process certificates from different CAs, the server uses a file called **certmap.conf**. This file contains instructions on how to interpret different certificates and how to search the directory for the information that those certificates contain.

In the Directory Server, a user entry has a format like the following:

```
dn: uid=jsmith,ou=People,dc=example,dc=com
...

cn: John Smith
mail: jsmith@example.com
```

A subject DN, however, is almost always formatted differently than an LDAP DN. For example:

```
cn=John Smith,e=jsmith@example.com,c=US,o=Example.com
```

The email attribute in the directory is almost always unique within the organization, as is the common name of the user. These attributes are also indexed by default, so they are easily searched, and are common attributes to be used in the subject names of certificates. The **certmap.conf** file can be configured so that the server looks for any mail or common name elements in the subject DN and matches them against the entries in the directory. Much like an **ldapsearch**, the cert mapping defines a search base (**DNComps**) and search filter (**FilterComps**).

```
certmap Example     o=Example.com,c=US
Example:DNComps     dc
Example:FilterComps mail,cn
```

The **certmap.conf** file is stored in the **/etc/dirsrv/slapd-*instance_name*/** folder. The file contains a default mapping as well as mappings for specific CAs.

The default mapping specifies what the server should do if a client certificate was issued by a CA that is not listed in **certmap.conf**. The mappings for specific CAs specify what the server should do for client certificates issued by those CAs. All mappings define the following:

- Where in the directory the server should begin its search
- What certificate attributes the server should use as search criteria
- Whether the server should verify the certificate with one that is stored in the directory

Mappings have the following syntax:

```
certmap name issuer DN
name:property [value]
name:property [value]
...
```

The first line of a mapping specifies the mapping's name as well as the DN for the issuer of the client certificate. The mapping can have any name, but the *issuerDN* must exactly match the issuer DN of the CA that issued the client certificate. For example, the following two *issuerDN* lines differ only in the number of spaces they contain, but the server would treat these two entries as different:

```
certmap moz ou=Example CA,o=Example,c=US
certmap moz ou=Example CA,o=Example,c=US
```

The second and subsequent lines of a mapping identify the rules that the server should use when searching the directory for information extracted from a certificate. These rules are specified through the use of one or more of the following properties:

- **DNComps**
- **FilterComps**
- **VerifyCert**

- **CmapLdapAttr**
- **Library**
- **InitFn**

DNComps

DNComps is a comma-separated list of relative distinguished name (RDN) keywords used to determine where in the user directory the server should start searching for entries that match the information for the owner of the client certificate. The server gathers values for these keywords from the client certificate and uses the values to form a DN, which determines where the server starts its search in the directory.

For example, if the **DNComps** is set to use the ***o*** and ***c*** RDN keywords, the server starts the search from the **o=***org*, **c=***country* entry in the directory, where *org* and *country* are replaced with values from the DN in the certificate.

- If there is not a **DNComps** entry in the mapping, the server uses either the **CmapLdapAttr** setting or the entire subject DN in the client certificate to determine where to start searching.
- If the **DNComps** entry is present but has no value, the server searches the entire directory tree for entries matching the filter specified by **FilterComps**.

The following RDN keywords are supported for **DNComps**:

- ***cn***
- ***ou***
- ***o***
- ***c***
- ***l***
- ***st***
- ***dc***
- ***e*** or ***mail*** (but not both)
- ***mail***

Keywords can be in either lower case or upper case.

FilterComps

FilterComps is a comma-separated list of RDN keywords used to create a filter by gathering information from the user's DN in the client certificate. The server uses the values for these keywords to form the search criteria for matching entries in the LDAP directory. If the server finds one or more entries in the directory that match the user's information gathered from the certificate, the search is successful and the server performs a verification (if **verifycert** is set to **on**).

For example, if **FilterComps** is set to use the ***e*** and ***uid*** attribute keywords (**FilterComps=e,uid**), the server searches the directory for an entry whose values for ***e*** and ***uid*** match the user's information gathered from the client certificate. Email addresses and user IDs are good filters because they are usually unique entries in the directory.

The filter needs to be specific enough to match one and only one entry in the directory.

The following RDN keywords are supported for **FilterComps**:

- ***cn***
- ***ou***
- ***o***
- ***c***
- ***l***
- ***st***
- ***dc***

- ***e*** or ***mail*** (but not both)
- ***mail***

Keywords can be in either lower case or upper case.

VerifyCert

verifycert tells the server whether it should compare the client's certificate with the certificate found in the user's directory entry. The value is either **on** or **off**. Setting the value to **on** ensures that the server will not authenticate the client unless the certificate presented exactly matches the certificate stored in the directory. Setting the value to **off** disables the verification process.

CmapLdapAttr

CmapLdapAttr is the name of the attribute in the directory that contains subject DNs from all certificates belonging to the user. Because this attribute is not a standard LDAP attribute, this attribute must be added to the schema. See Section 8.4.2, "Creating Attributes" for information on adding schema elements.

If the **CmapLdapAttr** property exists in a **certmap.conf** mapping, the server searches the entire directory for an entry that contains the subject's full DN. The search criteria are the attribute named by **CmapLdapAttr** and the subject's full DN as listed in the certificate. If the search does not yield any entries, the server retries the search using the **DNComps** and **FilterComps** mappings. The search will take place more quickly if the attribute specified by **CmapLdapAttr** is indexed. For more information on indexing attributes, see Chapter 9, *Managing Indexes*.

Using **CmapLdapAttr** to match a certificate to a directory entry is useful when it is difficult to match entries using **DNComps** and **FilterComps**.

Library

Library is the pathname to a shared library or DLL. Use this property only to extend or replace the standard functions that map information in **certmap.conf** to entries in the directory. This property is typically not necessary unless there are very specialized mapping requirements.

InitFn

InitFn is the name of an **init** function from a custom library. You need to use this property only if you want to extend or replace the functions that map information in **certmap.conf** to entries in the directory. This property is typically not necessary unless you have very specialized mapping requirements.

7.10.3. Editing the certmap.conf File

1. In a text editor, open **/etc/dirsrv/slapd-*instance_name*/certmap.conf**
2. If necessary, make changes to the default mapping.

   For example, change the value for **DNComps** or **FilterComps**. To comment out a line, insert a **#** before it.
3. If desired, create a mapping for a specific CA.

   The mapping should take the form **certmap** *mappingName issuerDN*.

   For example, to create a mapping named **Example CA** which has the issuer DN **ou=example CA,o=example,c=US**, enter the following:

   ```
   certmap example CA ou=example CA,o=example,c=US
   ```
4. Add property settings for a specific CA's mapping.

   Specify the **Library** and **InitFn** properties before adding any additional properties.

   When adding a property, use the form *mappingName:propertyName value*.

   For example, add a **DNComps** value of **o, c** for **Example CA** by entering the following line:

   ```
   example CA:DNComps o, c
   ```

   For the **Library** and **InitFn** properties, a complete mapping looks like this:

   ```
   certmap Example CA ou=example CA,o=example,c=US
   Example CA:Library /ldapserver/ldap/servers/slapd/plugin.c
   Example CA:InitFn plugin_init_dn
   ```

```
Example CA:DNComps o, c
Example CA:FilterComps e, uid
Example CA:VerifyCert on
Example CA:CmapLdapAttr certSubjectDN
```

5. Save the **certmap.conf** file.

7.10.4. Example certmap.conf Mappings

In Example 7.1, "Default Mapping", the server starts its search at the directory branch point containing the entry **ou=***organizationalUnit*, **o=***organization*, **c=***country*, where the italics represent values from the subject's DN in the client certificate.

Example 7.1. Default Mapping

```
certmap default     default
default:DNComps     ou, o, c
default:FilterComps e, uid
default:verifycert  on
```

The server then uses the values for ***e*** (email address) and ***uid*** (user ID) from the certificate to search for a match in the directory before authenticating the user. When it finds a matching entry, the server verifies the certificate by comparing the certificate the client sent to the certificate stored in the directory.

Example 7.2, "An Additional Mapping" shows the contents of a sample **certmap.conf** file that defines a default mapping as well as a mapping for MyCA:

Example 7.2. An Additional Mapping

```
certmap default     default
default:DNComps
default:FilterComps e, uid
certmap MyCA        ou=MySpecialTrust,o=MyOrg,c=US
MyCA:DNComps        ou,o,c
MyCA:FilterComps    e
MyCA:verifycert     on
```

When the server gets a certificate from a CA other than MyCA, the server uses the default mapping, which starts at the top of the directory tree and searches for an entry matching the client's email address (***e***) and user ID (***uid***). If the certificate is from MyCA, the server starts its search at the directory branch containing the organizational unit specified in the subject DN and searches for email addresses (***e***) that match the one specified in the certificate. If the certificate is from MyCA, the server verifies the certificate. If the certificate is from another CA, the server does not verify it.

Example 7.3, "A Mapping with an Attribute Search" uses the **CmapLdapAttr** property to search the directory for an attribute called **certSubjectDN** whose value exactly matches the entire subject DN in the client certificate:

Example 7.3. A Mapping with an Attribute Search

```
certmap MyCo        ou=My Company Inc,o=MyCo,c=US
MyCo:CmapLdapAttr   certSubjectDN
MyCo:DNComps        o, c
MyCo:FilterComps    mail, uid
MyCo:verifycert     on
```

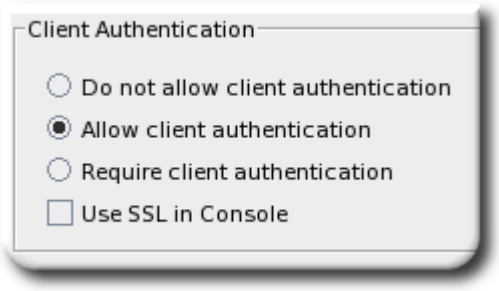
If the subject DN in the client certificate is **uid=jsmith,o=example Inc,c=US**, then the server searches for entries that have **certSubjectDN=uid=jsmith,o=example Inc,c=US**.

If one or more matching entries are found, the server proceeds to verify the entries. If no matching entries are found, the server uses **DNComps** and **FilterComps** to search for matching entries. For the client certificate described above, the server would search for **uid=jsmith** in all entries under **o=example Inc,c=US**.

7.10.5. Allowing and Requiring Client Authentication to the Console

Client authentication must be explicitly set in the Directory Server.

1. Click the **Configuration** tab.
2. With the top server entry highlighted in the left navigation pane, click the **Encryption** tab in the main window.
3. Set whether to require or allow client authentication to the Directory Server.



   - *Do not allow client authentication.* With this option, the server ignores the client's certificate. This does not mean that the bind will fail.
   - *Allow client authentication.* This is the default setting. With this option, authentication is performed on the client's request.
   - *Require client authentication.* With this option, the server requests authentication from the client.

     If client authentication is required, then SSL cannot be used with the Console because The Directory Server Console does not support client authentication.

NOTE

To use certificate-based authentication with replication, configure the consumer server either to allow or to require client authentication.

NOTE

The Directory Server must already be configured to run over TLS/SSL or Start TLS for client authentication to be enabled.

4. Save the changes, and restart the server. For example:

```
service dirsrv restart
```

To change the server configuration from requiring client authentication to allowing it through the command line, reset the ***nsSSLClientAuth*** parameter:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 636 -h
server.example.com -x

dn: cn=encryption,cn=config
changetype: modify
replace: nsSSLClientAuth
nsSSLClientAuth: allowed
```

The ***nsSSLClientAuth*** parameter can be set to **off**, **allowed**, and **required**.

7.10.6. Forcing SASL/EXTERNAL Mechanisms for Bind Requests

The assumption with certificate-based authentication is that the Directory Server will use the identity in the presented certificate to process the BIND request.

There are two steps that the client takes at the beginning of an SSL session to use certificate authentication. First, the client sends its certificate to the Directory Server. Then, it sends its bind request. Most clients issue the bind request using the SASL/EXTERNAL mechanism, which is important because it signals to the Directory Server that it needs to use the identity in the certificate for the bind, not any credentials in the bind request.

However, some clients use simple authentication or anonymous credentials when they send the bind request. The client still assumes that the Directory Server will use the identity in the certificate for the bind identity, but there is

nothing in the bind request to tell the Directory Server that is what it should do. In that case, the SSL session fails with invalid credentials, even if the certificate and the client identity in that certificate are valid.

A configuration option in the Directory Server (***nsslapd-force-sasl-external***) forces clients in certificate-based authentication to use the SASL/EXTERNAL method and to ignore any simple bind method given with the request.

This attribute is off by default, but it can be turned on:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 636 -h
server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-force-sasl-external
nsslapd-force-sasl-external: on
```

7.11. SETTING UP SASL IDENTITY MAPPING

Red Hat Directory Server supports LDAP client authentication through the Simple Authentication and Security Layer (SASL), an alternative to TLS/SSL and a native way for some applications to share information securely.

Simple Authentication and Security Layer (SASL) is an abstraction layer between protocols like LDAP and authentication methods like GSS-API which allows any protocol which can interact with SASL to utilize any authentication mechanism which can work with SASL. Simply put, SASL is an intermediary that makes authenticating to applications using different mechanisms easier. SASL can also be used to establish an encrypted session between a client and server.

The SASL framework allows different mechanisms to be used to authenticate a user to the server, depending on what mechanism is enabled in both client and server applications. SASL also creates a layer for encrypted (secure) sessions. Using GSS-API, Directory Server utilizes Kerberos tickets to authenticate sessions and encrypt data.

7.11.1. About SASL Identity Mapping

When processing a SASL bind request, the server matches, or maps, the SASL authentication ID used to authenticate to the Directory Server with an LDAP entry stored within the server. When using Kerberos, the SASL user ID usually has the format *userid@REALM*, such as **scarter@EXAMPLE.COM**. This ID must be converted into the DN of the user's Directory Server entry, such as **uid=scarter,ou=people,dc=example,dc=com**.

If the authentication ID clearly corresponds to the LDAP entry for a person, it is possible to configure the Directory Server to map the authentication ID automatically to the entry DN. Directory Server has some pre-configured default maps which handle most common configurations, and customized maps can be created. During a bind attempt, the first matching mapping rule is applied. If only one user identity is returned, the bind is successful; if none or more than one are returned, then the bind fails.

Be sure to configure SASL maps so that only one mapping rule matches the authentication string.

SASL mappings are configured by entries under a container entry:

```
dn: cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: sasl
```

SASL identity mapping entries are children of this entry:

```
dn: cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: mapping
```

Mapping entries are defined by three attributes:

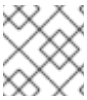
- *nsSaslMapRegexString*, the regular expression which is used to map the elements of the supplied ***authid***
- *nsSaslMapFilterTemplate*, a template which applies the elements of the ***nsSaslMapRegexString*** to create the DN
- *nsSaslMapBaseDNTemplate*, which provides the search base or a specific entry DN to match against the constructed DN

For example:

```
dn: cn=mymap,cn=mapping,cn=sasl,cn=config
objectclass:top
objectclass:nsSaslMapping
cn: mymap
nsSaslMapRegexString: \(.*\)@\(.*\)\.\(.*\)
nsSaslMapFilterTemplate: (objectclass=inetOrgPerson)
nsSaslMapBaseDNTemplate: uid=\1,ou=people,dc=\2,dc=\3
```

The ***nsSaslMapRegexString*** attribute sets variables of the form **\1**, **\2**, **\3** for bind IDs which are filled into the template attributes during a search. This example sets up a SASL identity mapping for any user in the **ou=People,dc=example,dc=com** subtree who belongs to the **inetOrgPerson** object class.

When a Directory Server receives a SASL bind request with **mconnors@EXAMPLE.COM** as the user ID (***authid***), the regular expression fills in the base DN template with **uid=mconnors,ou=people,dc=EXAMPLE,dc=COM** as the user ID, and authentication proceeds from there.



NOTE

The **dc** values are not case sensitive, so **dc=EXAMPLE** and **dc=example** are equivalent.

The Directory Server can also use a more inclusive mapping scheme, such as the following:

```
dn: cn=example map,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: example map
nsSaslMapRegexString: \(.*\)
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This matches any user ID and map it an entry under the **ou=People,dc=example,dc=com** subtree which meets the filter **cn=***userId*.

Mappings can be confined to a single realm by specifying the realm in the ***nsSaslMapRegexString*** attribute. For example:

```
dn: cn=example map,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: example map
nsSaslMapRegexString: \(.*\)@US.EXAMPLE.COM
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This mapping is identical to the previous mapping, except that it only applies to users authenticating from the **US.EXAMPLE.COM** realm. (Realms are described in Section 7.12.2.1, “About Principals and Realms”.)

When a server connects to another server, such as during replication or with chaining, the default mappings for the will not properly map the identities. This is because the principal (SASL identity) for one server does not match the principal on the server where authentication is taking place, so it does not match the mapping entries.

To allow server to server authentication using SASL, create a mapping for the specific server principal to a specific user entry. For example, this mapping matches the **ldap1.example.com** server to the **cn=replication manager,cn=config** entry. The mapping entry itself is created on the second server, such as **ldap2.example.com**.

```
dn: cn=z,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: z
nsSaslMapRegexString: ldap/ldap1.example.com@EXAMPLE.COM
nsSaslMapBaseDNTemplate: cn=replication manager,cn=config
nsSaslMapFilterTemplate: (objectclass=*)
```

Sometimes, the realm name is not included in the principal name in SASL GSS-API configuration. A second mapping can be created which is identical to the first, only without specifying the realm in the principal name. For example:

```
dn: cn=y,cn=mapping,cn=sasl,cn=config
```

```
objectclass: top
objectclass: nsSaslMapping
cn: y
nsSaslMapRegexString: ldap/ldap1.example.com
nsSaslMapBaseDNTemplate: cn=replication manager,cn=config
nsSaslMapFilterTemplate: (objectclass=*)
```

Because the realm is not specified, the second mapping is more general (meaning, it has the potential to match more entries than the first. The best practice is to have more specific mappings processed first and gradually progress through more general mappings.

There is no way to specify the order that mappings are processed. However, there is a way to control how SASL mappings are processed: the name. The Directory Server processes SASL mappings in reverse ASCII order. In the past two example, then the **cn=z** mapping (the first example) is processed first. If there is no match, the server processes the **cn=y** mapping (the second example).

NOTE

SASL mappings can be added when an instance is created during a silent installation by specifying the mappings in an LDIF file and adding the LDIF file with the **ConfigFile** directive. Using silent installation is described in the *Installation Guide*.

7.11.2. Default SASL Mappings for Directory Server

The Directory Server has pre-defined SASL mapping rules to handle some of the most common usage.

Kerberos UID Mapping

This matches a Kerberos principal using a two part realm, such as *user***@example.com**. The realm is then used to define the search base, and the user ID (***authid***) defines the filter. The search base is **dc=example,dc=com** and the filter of **(uid=user)**.

```
dn: cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: Kerberos uid mapping
nsSaslMapRegexString: \(.*\)@\(.*\)\.\(.*\)
nsSaslMapBaseDNTemplate: dc=\2,dc=\3
nsSaslMapFilterTemplate: (uid=\1)
```

RFC 2829 DN Syntax

This mapping matches an ***authid*** that is a valid DN (defined in RFC 2829) prefixed by **dn:**. The ***authid*** maps directly to the specified DN.

```
dn: cn=rfc 2829 dn syntax,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: rfc 2829 dn syntax
nsSaslMapRegexString: ^dn:\(.*\)
nsSaslMapBaseDNTemplate: \1
nsSaslMapFilterTemplate: (objectclass=*)
```

RFC 2829 U Syntax

This mapping matches an ***authid*** that is a UID prefixed by **u:**. The value specified after the prefix defines a filter of **(uid=value)**. The search base is hard-coded to be the suffix of the default **userRoot** database.

```
dn: cn=rfc 2829 u syntax,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: rfc 2829 u syntax
nsSaslMapRegexString: ^u:\(.*\)
nsSaslMapBaseDNTemplate: dc=example,dc=com
nsSaslMapFilterTemplate: (uid=\1)
```

UID Mapping

This mapping matches an ***authid*** that is any plain string that does not match the other default mapping rules. It use this value to define a filter of **(uid=value)**. The search base is hard-coded to be the suffix of the default **userRoot** database.

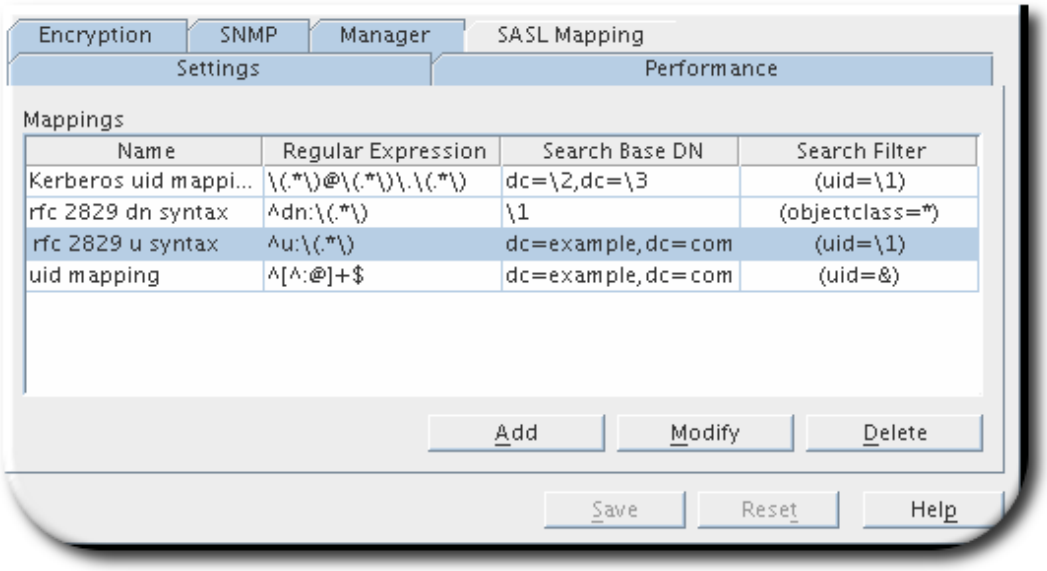
```
dn: cn=uid mapping,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: uid mapping
nsSaslMapRegexString: ^[^:@]+$
nsSaslMapBaseDNTemplate: dc=example,dc=com
nsSaslMapFilterTemplate: (uid=&)
```

7.11.3. Configuring SASL Identity Mapping

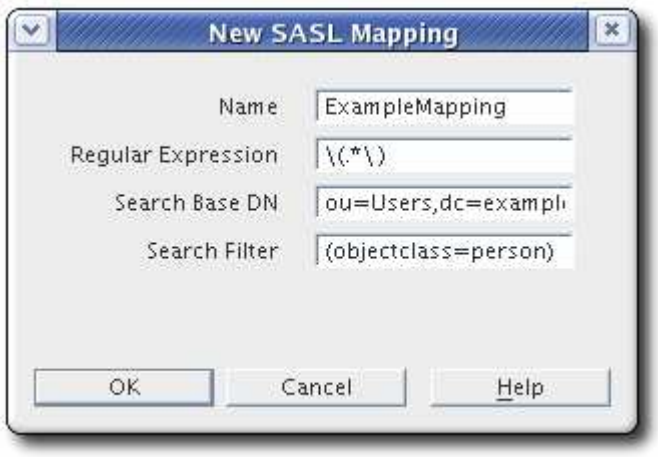
SASL identity mapping can be configured from either the Directory Server or the command line. For SASL identity mapping to work for SASL authentication, the mapping must return one, and only one, entry that matches and Kerberos must be configured on the host machine.

7.11.3.1. Configuring SASL Identity Mapping from the Console

1. In the Directory Server Console, open the **Configuration** tab.
2. Select the **SASL Mapping** tab.



3. To add a new SASL identity mapping, select the **Add** button, and fill in the required values.



- *Name.* This field sets the unique name of the SASL mapping.
- *Regular expression.* This field sets the regular expression used to match the DN components, such as **\(.*\)**. This field corresponds to the **nsSaslMapRegexString** value in the SASL mapping LDIF entry.
- *Search base DN.* This field gives the base DN to search to map entries, such as **ou=People,dc=example,dc=com**. This field corresponds to the **nsSaslMapBaseDNTemplate** value in the SASL mapping LDIF entry.

- *Search filter.* This field gives the search filter for the components to replace, such as **(objectclass=*)**. This field corresponds to the **nsSaslMapFilterTemplate** value in the SASL mapping LDIF entry.

To edit a SASL identity mapping, highlight that identity in the **SASL Mapping** tab, and click **Modify**. Change any values, and save.

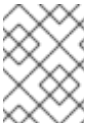
To delete a SASL identity mapping, highlight it and hit **Delete**. A dialog box comes up to confirm the deletion.

7.11.3.2. Configuring SASL Identity Mapping from the Command Line

To configure SASL identity mapping from the command line, use the **ldapmodify** utility to add the identity mapping scheme. For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example map,cn=mapping,cn=sasl,cn=config
changetype: add
objectclass: top
objectclass: nsSaslMapping
cn: example map
nsSaslMapRegexString: \(.*\)
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This matches any user's common name and maps it to the result of the subtree search with base **ou=People,dc=example,dc=com**, based on the filter **cn=***userId*.



NOTE

When SASL maps are added over LDAP, they are not used by the server until it is restarted. Adding the SASL map with **ldapmodify** adds the mapping to the end of the list, regardless of its ASCII order.

7.12. USING KERBEROS GSS-API WITH SASL

Kerberos v5 must be deployed on the host for Directory Server to utilize the GSS-API mechanism for SASL authentication. GSS-API and Kerberos client libraries must be installed on the Directory Server host to take advantage of Kerberos services.

7.12.1. Authentication Mechanisms for SASL in Directory Server

Directory Server support the following SASL encryption mechanisms:

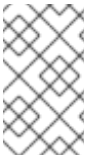
- *PLAIN.* PLAIN sends cleartext passwords for simple password-based authentication. This is not a preferred mechanism for most applications because of its relative lack of strength, but it can be used in some situations where anonymous access is disabled and an arbitrary UID (not a DN) is used to authenticate to the server because SASL can map the UID to a directory entry.
- *EXTERNAL.* EXTERNAL, as with TLS/SSL, performs certificate-based authentication. This method uses public keys for strong authentication.
- *CRAM-MD5.* **CRAM-MD5** is a simple challenge-response authentication method. It does not establish any security layer, unlike GSS-API. Both DIGEST-MD5 and GSS-API are much more secure, so both of those methods are recommended over CRAM-MD5.
- *DIGEST-MD5.* **DIGEST-MD5** is a mandatory authentication method for LDAPv3 servers. While it is not as strong as public key systems or Kerberos authentication methods, it is preferred over plain text passwords and does protect against plain text attacks.
- *Generic Security Services (GSS-API).* Generic Security Services (GSS) is a security API that is the native way for UNIX-based operating systems to access and authenticate Kerberos services. GSS-API also supports session encryption, similar to TLS/SSL. This allows LDAP clients to authenticate with the server using Kerberos version 5 credentials (tickets) and to use network session encryption.

  For Directory Server to use GSS-API, Kerberos must be configured on the host machine. See Section 7.12, "Using Kerberos GSS-API with SASL".

NOTE

GSS-API and, thus, Kerberos are only supported on platforms that have GSS-API support. To use GSS-API, it may be necessary to install the Kerberos client libraries; any required Kerberos libraries will be available through the operating system vendor.

CRAM-MD5, DIGEST-MD5, and GSS-API are all *shared secret* mechanisms. The server challenges the client attempting to bind with a *secret*, such as a password, that depends on the mechanism. The user sends back the response required by the mechanism.



NOTE

DIGEST-MD5 requires clear text passwords. The Directory Server requires the clear text password in order to generate the shared secret. Passwords already stored as a hashed value, such as **SHA1**, *cannot* be used with **DIGEST-MD5**.

7.12.2. About Kerberos in Directory Server

On Red Hat Enterprise Linux, the supported Kerberos libraries are MIT Kerberos version 5.

The concepts of Kerberos, as well as using and configuring Kerberos, are covered at the MIT Kerberos website, http://web.mit.edu/Kerberos/.

7.12.2.1. About Principals and Realms

A *principal* is a user in the Kerberos environment. A *realm* is a set of users and the authentication methods for those users to access the realm. A realm resembles a fully-qualified domain name and can be distributed across either a single server or a single domain across multiple machines. A single server instance can also support multiple realms.

NOTE

Kerberos realms are only supported for GSS-API authentication and encryption, not for DIGEST-MD5.

Realms are used by the server to associate the DN of the client in the following form, which looks like an LDAP DN:

```
uid=user_name/[server_instance],cn=realm,cn=mechanism,cn=auth
```

For example, Mike Connors in the **engineering** realm of the European division of **example.com** uses the following association to access a server in the US realm:

```
uid=mconnors/cn=Europe.example.com,cn=engineering,cn=gssapi,cn=auth
```

Babara Jensen, from the **accounting** realm of **US.example.com**, does not have to specify a realm when to access a local server:

```
uid=bjensen,cn=accounting,cn=gssapi,cn=auth
```

If realms are supported by the mechanism and the default realm is not used to authenticate to the server, then the *realm* must be specified in the Kerberos principal. Otherwise, the realm can be omitted.

NOTE

Kerberos systems treat the Kerberos realm as the default realm; other systems default to the server.

7.12.2.2. About the KDC Server and Keytabs

Kerberos is a protocol which allows users or servers to authenticate to a server securely over an insecure connection. As with protocols like TLS and SSL, Kerberos generates and issues session keys which encrypt information. A Kerberos server, then, has two functions: as an authenticating server to validate clients and as a ticket granting server.

Because of this, the Kerberos server is called a *key distribution center* or KDC.

See the operating system documentation for information on installing and configuring a Kerberos server.

When a client authenticates to the Directory Server using GSS-API, the KDC sends a session key, followed by a ticket granting ticket (TGT). The TGT contains the client's ID and network address, a validity period, and the session key. The ticket and the ticket's lifetime are parameters in the Kerberos client and server configuration. In many systems,

this TGT is issued to the client when the user first logs into the operating system.

Command-line utilities provided with the operating system – including **kinit**, **klist**, and **kdestroy** – can acquire, list, and destroy the TGT. These tools usually use a file called a *keytab* to the issued keys. This file is created by the Kerberos administrator by exporting the key from the KDC.

In order to respond to Kerberos operations, the Directory Server requires access to its own cryptographic key. The Kerberos key is written in the keytab file. The keytab gives the credentials that the Directory Server uses to authenticate to other servers. Directory Server assigns a keytab through the **KRB5_KTNAME** environment variable in its startup script (**/etc/sysconfig/dirsrv**):

```
KRB5_KTNAME=/etc/dirsrv/ds.keytab ; export KRB5_KTNAME
```

NOTE

The Directory Server must be able to access the host keytab and the **krb5.conf** file for GSS-API authentication in SASL. For this host keytab file to be properly labeled in SELinux in the **dirsrv_config_t** context, the file must be in the **/etc** directory.

Only the host keytab and **krb5.conf** must be in **/etc**. The user key tabs can still be in any directory.

When the Directory Server authenticates to the server, it is as if it is running a **kinit** command to initiate the connection:

```
kinit -k -t /etc/dirsrv/ds.keytab ldap/FQDN@REALM
```

The Directory Server uses the service name **ldap**. Its Kerberos principal is **ldap**/*host-fqdn*@*realm*, such as **ldap/ldap.corp.example.com/EXAMPLE.COM**. The *host-fqdn* must be the fully-qualified host and domain name, which can be resolved by all LDAP and Kerberos clients through both DNS and reverse DNS lookups. A key with this identity must be stored in the server's keytab in order for Kerberos to work.

Whatever server the Directory Server is authenticating to must have a SASL mapping that maps the Directory Server's principal (its user entry, usually something like **ldap/server.example.com@EXAMPLE.COM**) to a real entry DN in the receiving server.

On Red Hat Enterprise Linux, the client-side Kerberos configuration is in the **/etc/krb5.conf**.

NOTE

The Directory Server must be able to access the **krb5.conf** file for GSS-API authentication in SASL. For these files to be properly labeled in SELinux in the **dirsrv_config_t** context, the file must be in the **/etc** directory.

For information on setting up the service key, see the Kerberos documentation.

Example 7.4, "KDC Server Configuration" shows a KDC server configured with the **company.example.com** realm.

Example 7.4. KDC Server Configuration

```
[libdefaults]
    ticket_lifetime = 24000
    default_realm = COMPANY.EXAMPLE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    ccache_type = 1
    forwardable = true
    proxiable = true
    default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc
    default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
    permitted_enctypes = des3-hmac-sha1 des-cbc-crc
[realms]
  COMPANY.EXAMPLE.COM = {
    kdc = kdcserver.company.example.com:88
    admin_server =adminserver.company.example.com:749
    default_domain = company.example.com
  }
[appdefaults]
  pam = {
```

```
    debug = true
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
    krb4_convert = false
  }
[logging]
  default = FILE:/var/krb5/kdc.log
  kdc = FILE:/var/krb5/kdc.log
  admin_server = FILE:/var/log/kadmind.log
```

7.12.3. Configuring SASL Authentication at Directory Server Startup

SASL GSS-API authentication has to be activated in Directory Server so that Kerberos tickets can be used for authentication. This is done by supplying a system configuration file for the init scripts to use which identifies the variable to set the keytab file location. When the init script runs at Directory Server startup, SASL authentication is then immediately active.

The default SASL configuration file is in **/etc/sysconfig/dirsrv**.

If there are multiple Directory Server instances and not all of them will use SASL authentication, then there can be instance-specific configuration files created in that directory named **dirsrv-***instance*. For example, **dirsrv-example**. The default **dirsrv** file can be used if there is a single instance on a host.

To enable SASL authentication, uncomment the **KRB5_KTNAME** line in the **/etc/sysconfig/dirsrv** (or instance-specific) file, and set the keytab location for the **KRB5_KTNAME** variable. For example:

```
# In order to use SASL/GSSAPI the directory
# server needs to know where to find its keytab
# file - uncomment the following line and set
# the path and filename appropriately
KRB5_KTNAME=/etc/dirsrv/krb5.keytab ; export KRB5_KTNAME
```

7.12.4. Using an External Keytab

A default keytab file is specified in the Directory Server start script and is used by the Directory Server automatically. However, it is possible to specify a different keytab file, referencing a different principal, by manually running **kinit** and then specifying the cached credentials.

To specify the cached **kinit** credentials, add the principal as the **KRB5CCNAME** line in **/etc/sysconfig/dirsrv**:

```
KRB5CCNAME=/tmp/krb_ccache ; export KRB5CCNAME
kinit principalname
# how to provide the password here is left as an exercise
# or kinit -k -t /path/to/file.keytab principalname
chown serveruid:serveruid $KRB5CCNAME
# so the server process can read it
# start a cred renewal "daemon"
( while XXX ; do sleep NNN ; kinit ..... ; done ) &
# the exit condition XXX and sleep interval NNN are left as an exercise
...
```

The server has no way to renew these cached credentials. The **kinit** process must be run manually, external to Directory Server processes, or the server could begin receiving SASL bind failures when the server attempts to use expired credentials.

7.13. DISABLING SASL MECHANISMS

The root dse attribute ***supportedSASLMechanisms*** lists the SASL mechanisms that are *currently supported* by the Directory Server instance. However, editing that attribute does not change which mechanisms are supported. Directory Server uses the installed Cyrus SASL libraries to generate the list of supported SASL mechanisms. These libraries are located in **/usr/lib[64]/sasl2/**.

To change the list of SASL mechanisms supported by Directory Server:

1. Create a private SASL directory for the Directory Server instance to use. For example:

   ```
   mkdir /etc/dirsrv/slapd-instance_name/sasl2
   ```

2. Open that directory.
3. Create symlinks from the Cyrus SASL directory plug-ins to the instance directory. For example:

```
[root@server ~]# cd /etc/dirsrv/slapd-instance_name/sasl2 ; for file in /usr/lib64/sasl2/*.so* ; do
  ln -s $file
done
```

4. Remove the symlinks for the mechanisms that should *not* be supported in the Directory Server instance. For example:

```
rm *cram*
```

5. Edit the Directory Server start shell script so that it uses the Directory Server instance's SASL directory.

```
vim  /usr/lib[64]/dirsrv/slapd-example/start-slapd

SASL_PATH=/etc/dirsrv/slapd-instance_name/sasl2 ; export SASL_PATH
```

6. Restart the Directory Server.

```
service dirsrv restart
```

NOTE

Cyrus SASL can set a specific list of mechanisms to use for different applications within its own configuration, in a **/usr/lib/sasl/***appName***.conf** file.

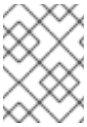
For more information, see the Cyrus SASL administrator's documentation at http://cyrusimap.web.cmu.edu/docs/cyrus-sasl/1.5.28/sysadmin.php.

7.14. USING SASL WITH LDAP CLIENTS

Directory Server uses SASL as an alternative TLS/SSL, particularly for environments which are using Kerberos to implement single sign-on. Directory Server allows user to use SASL to authenticate and bind to the server. This includes LDAP tools like **ldapsearch** and **ldapmodify**. For example:

```
ldapsearch -O noplain,minssf=1,maxbufsize=512 -Y GSSAPI -U "dn:uid=jsmith,ou=people,dc=example,dc=com" -
R EXAMPLE.COM ...
```

Using SASL with the LDAP client tools is described in Section A.3, “Using SASL with LDAP Client Tools” .



NOTE

SASL proxy authorization is not supported in Directory Server; therefore, Directory Server ignores any SASL **authzid** value supplied by the client.

[2] While SSLv2 *can* be enabled, SSLv2 is disabled by default in Directory Server and generally does not need to be enabled.

CHAPTER 8. MANAGING THE DIRECTORY SCHEMA

Red Hat Directory Server comes with a standard schema that includes hundreds of object classes and attributes. While the standard object classes and attributes should meet most deployments' requirements, it can be necessary to extend the schema for specific directory data. Extending the schema is done by creating new object classes and attributes.

The *Red Hat Directory Server 9 Configuration, Command, and File Reference* is a reference for most the standard Directory Server attributes and object classes, with information on allowed and required attributes, which object classes take which attribute, and OID and value information. This is a good resource for identifying useful schema elements for a directory and determining what custom schema needs to be created.

8.1. OVERVIEW OF SCHEMA

The directory schema is a set of rules that defines how data can be stored in the directory. Directory information is stored discrete entries, and each entry is comprised of a set of attributes and their values. The kind of identity being described in the entry is defined in the entry's object classes. An object class specifies the kind of object the entry describes through the defined set of attributes for the object class.

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, including people, groups, locations, organizations and divisions, and equipment. The identity is described in a directory entries with attributes and their values, pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. Other aspects of the Directory Server configuration, including matching rules and LDAP controls, are also defined in the schema. All of these together are *schema elements*.

Every schema element is identified by a unique, dot-separated number. This is called the *object identifier* or *OID*.

8.1.1. Default Schema Files

The schema for Directory Server is defined in several different schema files (LDIF files which define schema elements). The Directory Server schema files are instance-specific and are located in the **/etc/dirsrv/slapd-*instance_name*/schema** directory. There is also a common **/etc/dirsrv/schema** directory; the files in this directory are used as templates for new Directory Server instances. Putting custom schema in the **/etc/dirsrv/schema** directory means that it is automatically included with any new instances.

The default schema files are listed and described in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*, which also describes the common standard attributes and object classes as well as the attributes used by the Directory Server to perform operations and manage entries.

8.1.2. Object Classes

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, such as people (**person** and **inetOrgPerson**), groups (**groupOfUniqueNames**), locations (**locality**), organizations and divisions (**organization** and **organizationalUnit**), and equipment (**device**).

In a schema file, an object class is identified by the ***objectclasses*** line, then followed by its OID, name, a description, its direct superior object class (an object class which is required to be used in conjunction with the object class and which shares its attributes with this object class), and the list of required (**MUST**) and allowed (**MAY**) attributes.

This is shown in Example 8.1, “person Object Class Schema Entry” .

Example 8.1. person Object Class Schema Entry

```
objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard LDAP objectclass' SUP top MUST ( sn $ cn ) MAY (
description $ seeAlso $ telephoneNumber $ userPassword ) X-ORIGIN 'RFC 2256' )
```

Every object class defines a number of required attributes and of allowed attributes. Required attributes must be present in entries using the specified object class, while allowed attributes are permissible and available for the entry to use, but are not required for the entry to be valid.

As in Example 8.1, “person Object Class Schema Entry” , the **person** object class requires the ***cn***, ***sn***, and ***objectClass*** attributes and allows the ***description***, ***seeAlso***, ***telephoneNumber***, and ***userPassword*** attributes.

An object class can inherit attributes from another class, in addition to its own required and allowed attributes. The second object class is the *superior* or *parent* object class of the first.

For example, a user's entry has to have the **inetOrgPerson** object class. In that case, the entry must also include the superior object class for **inetOrgPerson**, **organizationalPerson**, and the superior object class for **organizationalPerson**, which is **person**:

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

8.1.3. Attributes

Directory entries are composed of attributes and their values. These pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. For instance, the ***cn*** attribute is used to store a person's full name, such as **cn: John Smith**.

Additional attributes can supply additional information about John Smith:

```
givenname: John
surname: Smith
mail: jsmith@example.com
```

In a schema file, an attribute is identified by the ***attributetypes*** line, then followed by its OID, name, a description, syntax (allowed format for its value), optionally whether the attribute is single- or multi-valued, and where the attribute is defined.

This is shown in Example 8.2, “description Attribute Schema Entry” .

Example 8.2. description Attribute Schema Entry

```
attributetypes: ( 2.5.4.13 NAME 'description' DESC 'Standard LDAP attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'RFC 2256' )
```

8.1.4. Extending the Schema

New, custom attributes and object classes can be added to a Directory Server instance to extend the schema, and there are several ways to add schema elements. Using the Directory Server Console or LDAP tools adds schema elements to the default custom schema file for an instance, **99user.ldif**. It is also possible to create a new, separate schema file and include it with the default schema files.

Adding new schema elements requires three things:

1. Planning and defining OIDs for the new schema. Schema elements are recognized by the server by their OID, so it is important for the OIDs to be unique and organized. Directory Server itself does not manage OIDs, but there are some best practices described in Section 8.2, “Managing Object Identifiers” .

2. Create the new attributes. Attribute definitions require a name, a syntax (the allowed format of the values), an OID, and a description of whether the attribute can only be used once per entry or multiple times.

3. Create an object class to contain the new attributes. An object class lists the required attributes for that entry type and the allowed (permissible) attributes. Because the default schema should never be altered, if any new attributes are created, then they should be added to a custom object class.

The schema elements should be planned in advance; do not use multiple attributes for the same information. Whenever possible, use the standard Directory Server schema. Directory Server has hundreds of attributes and dozens of object classes defined in the default schema files. The *Red Hat Directory Server 9 Configuration, Command, and File Reference* lists and describes the standard attributes and object classes; all of the schema can be viewed in the Directory Server Console or read in the schema files in **/etc/dirsrv/slapd-*instance_name*/schema**. Become familiar with the available schema; then plan what information attributes are missing and how best to fill those gaps with custom attributes. Planning the schema is covered in the *Deployment Guide*.

WARNING

The default object classes and attributes in Directory Server are based on LDAP and X.500 standards and RFCs. Using standard schema makes the Directory Server more easily integrated with other applications and servers and allows interoperability with LDAP clients, legacy Directory Server instances, and future release. It is inadvisable for you to edit the standard attributes or change the object classes.

Keep the following rules in mind when customizing the Directory Server schema:

- Keep the schema as simple as possible.
- Reuse existing schema elements whenever possible.
- Minimize the number of mandatory attributes defined for each object class.
- Do not define more than one object class or attribute for the same purpose.
- Do not modify any existing definitions of attributes or object classes.

NOTE

Never delete or replace the standard schema. Doing so can lead to compatibility problems with other directories or other LDAP client applications.

The schema is loaded into the Directory Server instance when the instance is started; any new schema files are not loaded until the Directory Server is restarted or unless a reload task is initiated. Always, any custom schema is loaded before the standard schema.

8.1.5. Schema Replication

When the directory schema is updated in the **cn=schema** sub-tree, Directory Server stores the changes in the local **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** file, including a change state number (CSN). The updated schema is not automatically replicated to other replicas. The schema replication starts when directory content is updated in the replicated tree. For example, if you update a user or group entry after modifying the schema, the supplier compares the CSN stored in the ***nsSchemaCSN*** attribute with the one on the consumer. If the remote CSN is lower than the one on the supplier, the schema is replicated to the consumer. For a successful replication, all object classes and attribute types on the supplier must be a superset of the consumer's definition.

Example 8.3. Schema subsets and supersets

- On **server1**, the **demo** object class allows the ***a1***, ***a2***, and ***a3*** attributes.
- On **server2**, the **demo** object class allows the ***a1*** and ***a3*** attributes.

In Example 8.3, "Schema subsets and supersets" , the schema definition of the **demo** object class on **server1** is a superset of the object class on **server2**. During the validation phase, when the schema is being replicated or accepted, Directory Server retrieves the superset definitions. For example, if a consumer detects that an object class in the local schema allows less attributes than the object class in the supplier schema, the local schema is updated.

If the schema definitions are successfully replicated, the ***nsSchemaCSN*** attributes are identical on both servers and no longer compared at the beginning of a replication session.

In the following scenarios, the schema is not replicated:

- The schema on one host is a subset of the schema of another host.

  For example, in Example 8.3, "Schema subsets and supersets" , the schema definition of the **demo** object class on **server2** is a subset of the object class on **server1**. Subsets can also occur for attributes (a single-value attribute is a subset of a multi-value attribute) and attribute syntaxes (**IA5** is a subset of **Octet_string**).

- When definitions in supplier schema and consumer schema need to be merged.

Directory Server does not support merging schemas. For example, if an object class on one server allows the ***a1***, ***a2***, and ***a3*** attributes and ***a1***, ***a3***, and ***a4*** on the other, the schemas are not subsets and cannot be merged.

- Schema files other than **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** are used.

  Directory Server enables you to add additional schema files in the **/etc/dirsrv/slapd-*instance_name*/schema/** directory. However, only the CSN in the **99user.ldif** file is updated. For this reasons, other schema file are only used locally and are not automatically transferred to replication partners. Copy the updated schema file manually to the consumers and reload the schema. For details, see Section 8.7, “Dynamically Reloading Schema”.

  To avoid duplicate schema definitions and to enable automatic replication, store all custom schema in the **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** file. For further information about creating custom schema files, see Section 8.6, “Creating Custom Schema Files”.

8.2. MANAGING OBJECT IDENTIFIERS

Each LDAP object class or attribute must be assigned a unique name and *object identifier* (OID). An OID is a dot-separated number which identifies the schema element to the server. OIDs can be hierarchical, with a base OID that can be expanded to accommodate different branches. For example, the base OID could be **1**, and there can be a branch for attributes at **1.1** and for object classes at **1.2**.

NOTE

It is not required to have a numeric OID for creating custom schema, but Red Hat strongly recommends it for better forward compatibility and performance.

OIDs are assigned to an organization through the Internet Assigned Numbers Authority (IANA), and Directory Server does not provide a mechanism to obtain OIDs. To get information about obtaining OIDs, visit the IANA website at http://www.iana.org/cgi-bin/enterprise.pl.

After obtaining a base OID from IANA, plan how the OIDs are going to be assigned to custom schema elements. Define a branch for both attributes and object classes; there can also be branches for matching rules and LDAP controls.

Once the OID branches are defined, create an OID registry to track OID assignments. An OID registry is a list that gives the OIDs and descriptions of the OIDs used in the directory schema. This ensures that no OID is ever used for more than one purpose. Publish the OID registry with the custom schema.

8.3. DIRECTORY SERVER ATTRIBUTE SYNTAXES

The attribute's syntax defines the format of the values which the attribute allows; as with other schema elements, the syntax is defined for an attribute using the syntax's OID in the schema file entry. In the Directory Server Console, the syntax is referenced by its friendly name.

The Directory Server uses the attribute's syntax to perform sorting and pattern matching on entries.

For more information about LDAP attribute syntaxes, see RFC 4517.

Table 8.1. Supported LDAP Attribute Syntaxes

Name	OID	Definition
Binary	1.3.6.1.4.1.1466.115.121.1.5	*Deprecated. Use Octet string instead.*
Bit String	1.3.6.1.4.1.1466.115.121.1.6	For values which are bitstings, such as **'0101111101'B**.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	For attributes with only two allowed values, TRUE or FALSE.
Country String	1.3.6.1.4.1.1466.115.121.1.11	For values which are limited to exactly two printable string characters; for example, US for the United States.
DN	1.3.6.1.4.1.1466.115.121.1.12	For values which are distinguished names (DNs).

Name	OID	Definition
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	For values which are contained a preferred method of delivering information or contacting an entity. The different values are separated by a dollar sign ($). For example: `telephone $ physical`
Directory String	1.3.6.1.4.1.1466.115.121.1.15	For values which are valid UTF-8 strings. These values are not necessarily case-insensitive. Both case-sensitive and case-insensitive matching rules are available for Directory String and related syntaxes.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	For values which contain complex search parameters based on attributes and filters.
Facsimile	1.3.6.1.4.1.1466.115.121.1.22	For values which contain fax numbers.
Fax	1.3.6.1.4.1.1466.115.121.1.23	For values which contain the images of transmitted faxes.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	For values which are encoded as printable strings. The time zone must be specified. It is strongly recommended to use GMT time.
Guide	1.3.6.1.4.1.1466.115.121.1.25	*Obsolete.* For values which contain complex search parameters based on attributes and filters.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	For values which are valid strings. These values are not necessarily case-insensitive. Both case-sensitive and case-insensitive matching rules are available for IA5 String and related syntaxes.
Integer	1.3.6.1.4.1.1466.115.121.1.27	For values which are whole numbers.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	For values which contain image data.
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	For values which contain a combination value of a DN and (optional) unique ID.
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	For values which contain a string of both numerals and spaces.
OctetString	1.3.6.1.4.1.1466.115.121.1.40	For values which are binary; this replaces the binary syntax.
OID	1.3.6.1.4.1.1466.115.121.1.37	For values which contain an object identifier (OID).

Name	OID	Definition
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	For values which are encoded in the format **postal-address =** *dstring* * **("$"** *dstring***)**. For example: 1234 Main St.$Raleigh, NC 12345$USA Each *dstring* component is encoded as a DirectoryString value. Backslashes and dollar characters, if they occur, are quoted, so that they will not be mistaken for line delimiters. Many servers limit the postal address to 6 lines of up to thirty characters.
PrintableString	1.3.6.1.4.1.1466.115.121.1.58	For values which contain strings containing alphabetic, numeral, and select punctuation characters (as defined in RFC 4517).
Space-Insensitive String	2.16.840.1.113730.3.7.1	For values which contain space-insensitive strings.
TelephoneNumber	1.3.6.1.4.1.1466.115.121.1.50	For values which are in the form of telephone numbers. It is recommended to use telephone numbers in international form.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	For values which contain an international telephone number.
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	For values which contain a telex number, country code, and answerback code of a telex terminal.
URI		For values in the form of a URL, introduced by a string such as **http://**, **https://**, **ftp://**, **ldap://**, and **ldaps://**. The URI has the same behavior as IA5 String. See RFC 4517 for more information on this syntax.

8.4. MANAGING CUSTOM SCHEMA IN THE CONSOLE

The Directory Server Console shows all attributes in the schema, and custom attributes can be created, edited, and deleted from the schema.

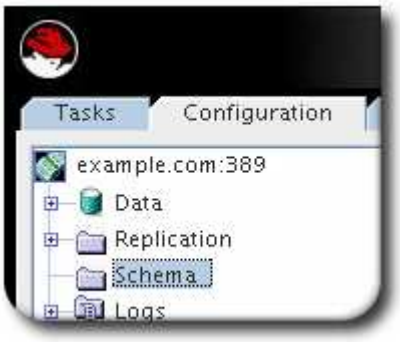
- Section 8.4.1, “Viewing Attributes and Object Classes”
- Section 8.4.2, “Creating Attributes”
- Section 8.4.3, “Creating Object Classes”
- Section 8.4.4, “Editing Custom Schema Elements”
- Section 8.4.5, “Deleting Schema”

8.4.1. Viewing Attributes and Object Classes

All of the information about the attributes and object classes which are currently loaded in the server instance are visible with the other server configuration.

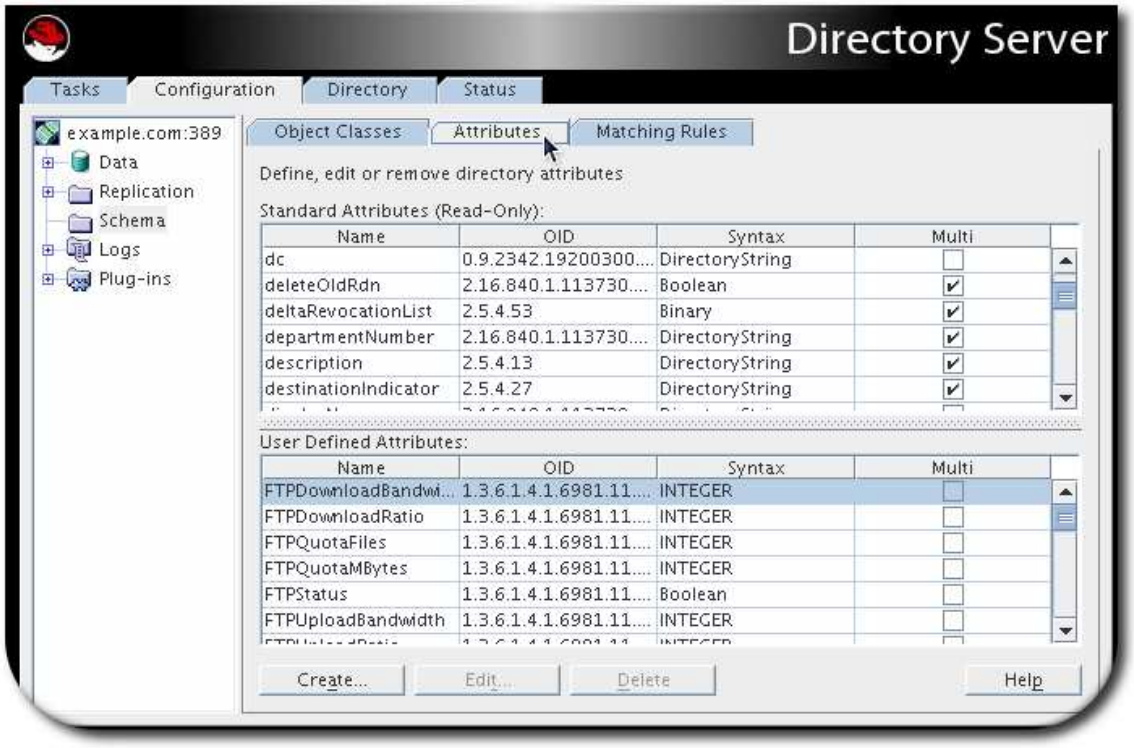
1. In the Directory Server Console, select the **Configuration** tab.

2. In the left navigation tree, select the **Schema** folder.

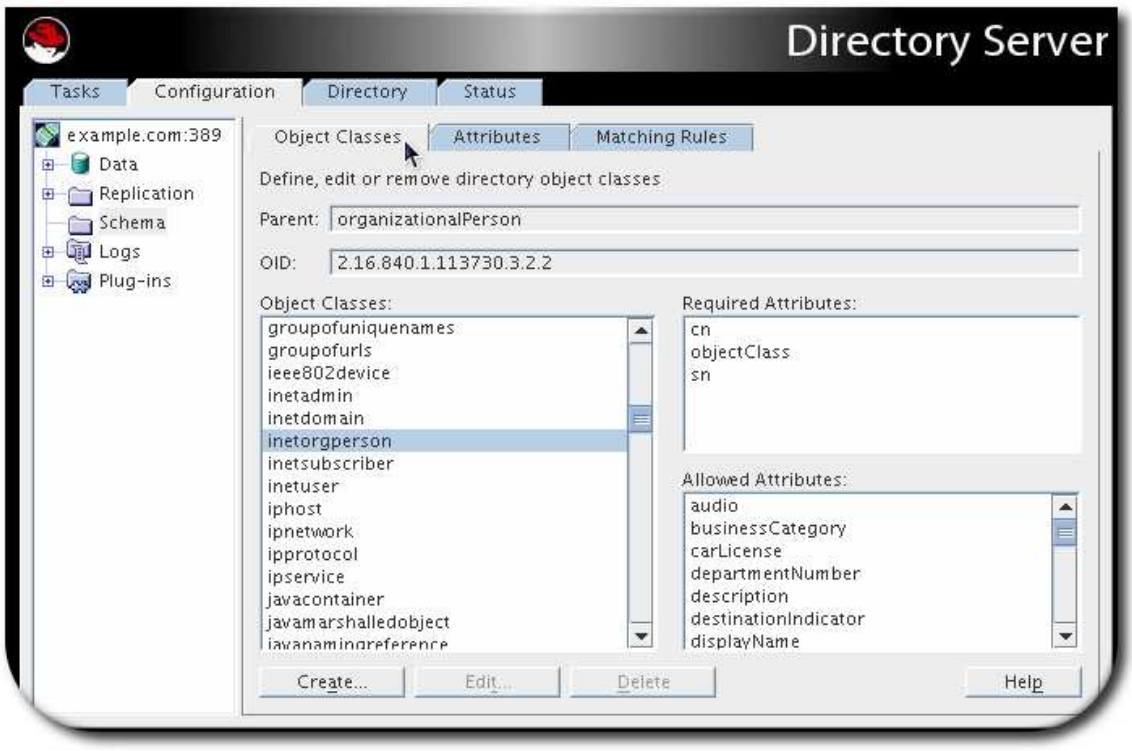


3. There are three tabs which display the schema elements loaded in Directory Server: **Object Class**, **Attributes**, and **Matching Rules**.

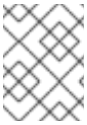
The **Attributes** tab is broken into two sections for default and custom attributes. Both sections show the attribute name, OID, syntax, and whether the attribute is multi-valued.



The **Object Classes** tab shows the list of object classes on the left. When an object class is highlighted, its OID and superior object class are listed in the fields at the top and its required and allowed attributes are listed in the boxes on the right.



8.4.2. Creating Attributes

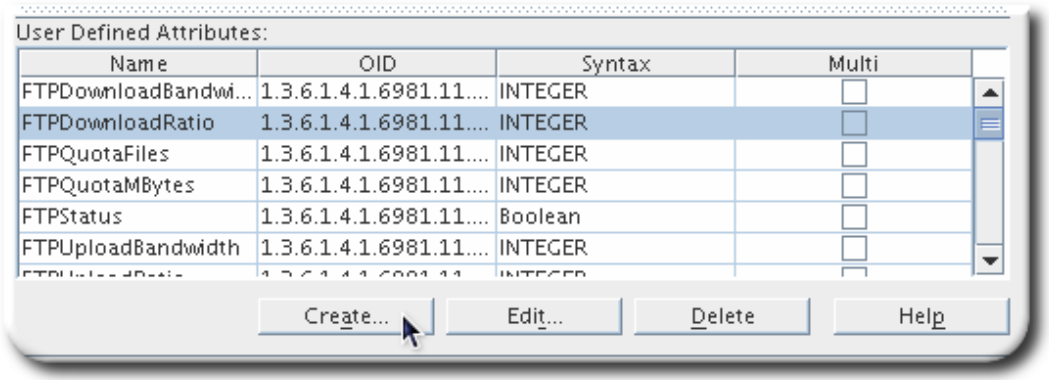


NOTE

After adding new attributes to the schema, create a new object class to contain them, as described in Section 8.4.3, "Creating Object Classes".

1. Select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder, and then select the **Attributes** tab in the right pane.

3. Click **Create**.



4. Fill in the information for the new attribute.

   - The attribute name; this must be unique.
   - The OID; this is not required, but for compatibility and server performance, assigning a unique numeric OID is strongly recommended.
   - The syntax; this is the allowed format for the attributes values.
   - Whether the attribute is multi-valued; by default, all attributes can be used more than once in an entry, but deselecting the check box means the attribute can be used only once.

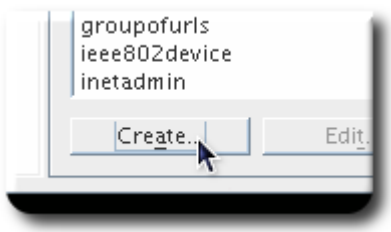
5. Click **OK**.

8.4.3. Creating Object Classes

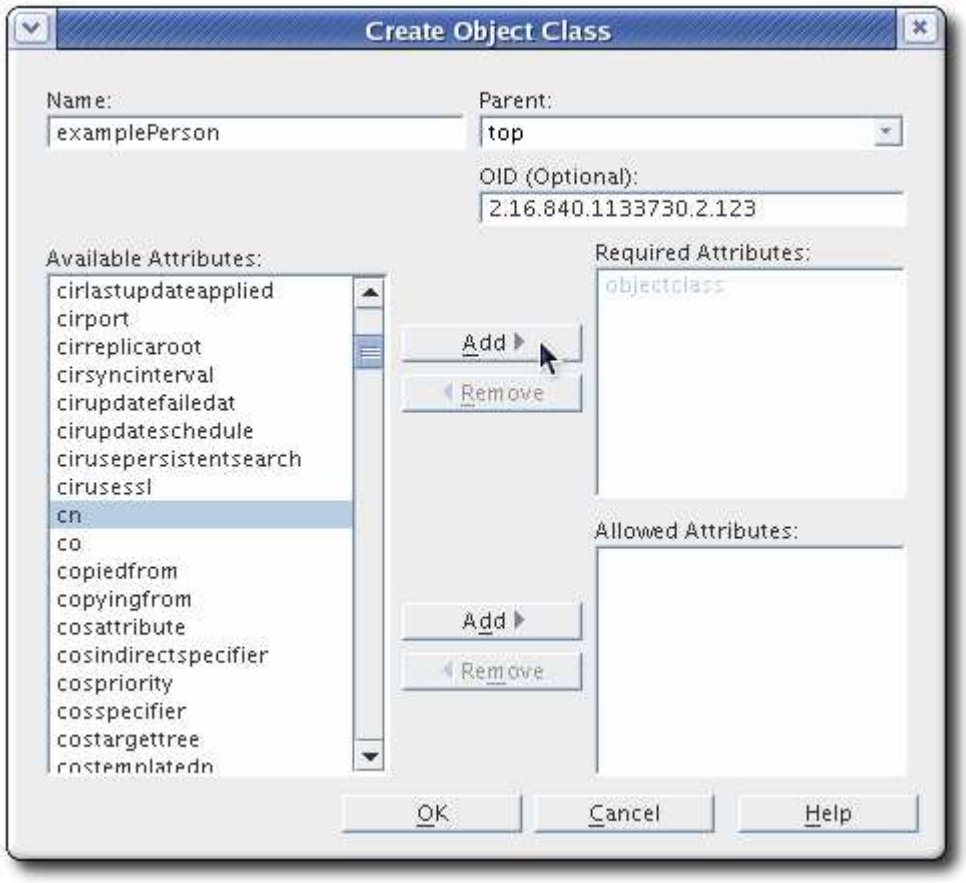
A new object class must be created with a unique name, a parent object, and required and optional attributes. To create an object class:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.

3. Click the **Create** button in the **Object Classes** tab.



4. Fill in the information about the new object class.



- The name; this must be unique.
- The OID; this is not required, but for compatibility and server performance, assigning a unique numeric OID is strongly recommended.
- The superior object class for the entry. The default is **top**; selecting another object class means that the new object class inherits all of the required and allowed attributes from the parent, in addition to its own defined attributes.
- Required and allowed attributes. Select the attributes on the left and used the **Add** buttons by the **Available Attributes** and **Required Attributes** boxes to add the attributes as appropriate.

NOTE

Attributes that are inherited from the parent object classes cannot be removed, regardless of whether they are allowed or required.

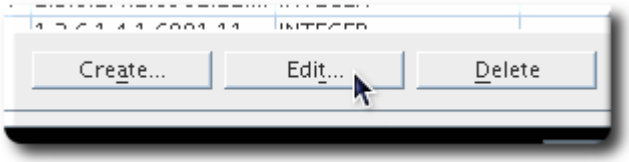
5. Click **OK** to save the new object class.

8.4.4. Editing Custom Schema Elements

Only user-created attributes or object classes can be edited; standard schema elements cannot be edited.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder.

3. Open the **Object Classes** or **Attributes** tab.
4. Select the schema element to edit from the list. Only custom (user-defined) schema can be edited in the Directory Server Console.
5. Click the **Edit** button at the bottom of the window.

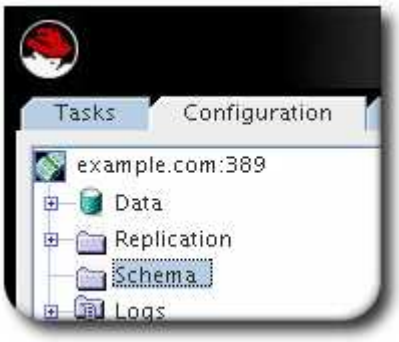


6. Edit any of the schema information.

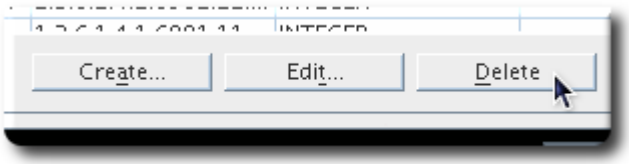
8.4.5. Deleting Schema

Only user-created attributes or object classes can be deleted; standard schema elements cannot be deleted.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder.



3. Open the **Object Classes** or **Attributes** tab.
4. Select the schema element to delete from the list. Only custom (user-defined) schema can be deleted in the Directory Server Console.
5. Click the **Delete** button at the bottom of the window.



6. Confirm the deletion.

WARNING

The server immediately deletes the schema element. There is no undo.

8.5. MANAGING SCHEMA USING LDAPMODIFY

As with the Directory Server Console, **ldapmodify** can be used to add, edit, and delete custom schema elements. **ldapmodify** also modifies the default custom schema file for a Directory Server instance, **99user.ldif**.

8.5.1. Creating Attributes

A custom attribute entry is itself an ***attributetypes*** entry for the **cn=schema** entry. The ***attributetypes*** attribute has the format:

```
attributetypes: ( definition )
```

The definition contains five components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The OID for the syntax of the attribute values, listed in Table 8.1, "Supported LDAP Attribute Syntaxes", in the form **SYNTAX** *OID*
- Optionally, the source where the attribute is defined

The attribute definition is added to the custom schema file, **99user.ldif**, by by running an LDAP command and modifying the **cn=schema** entry. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -v

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.2.3.4.5.6.1 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUED X-ORIGIN 'Example defined')
```

8.5.2. Creating Object Classes

An object class definition is an ***objectclasses*** attribute for the **cn=schema** entry. The ***objectclasses*** attribute has the format:

```
objectclasses: ( definition )
```

The object class definition contains several components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The superior, or parent, object class for this object class, in the form **SUP** *object_class*; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry
- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)

- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)

The object class definition is added to the custom schema file, **99user.ldif**, by by running an LDAP command and modifying the **cn=schema** entry. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x -v

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object Class' SUP
inetOrgPerson AUXILIARY  MUST cn MAY (exampleDateOfBirth $ examplePreferredOS) )
```

8.5.3. Deleting Schema

WARNING

Never delete default schema elements. Those are required by the Directory Server to run.

1. Remove the unwanted attributes from any entries which use them, then from any object classes in the schema file which accept that attribute. Likewise, to remove an object class, remove it from any entries.
2. Run **ldapmodify** to remove the attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object Class'
SUP inetOrgPerson AUXILIARY MUST cn MAY (exampleDateOfBirth $ examplePreferredOS) )
```

WARNING

Be sure to specify the exact object class or attribute to remove; using only the ***attributetypes*** or ***objectclasses*** attribute without the value will delete every user-defined attribute or object class in the file.

If the custom attribute or object class is in a custom schema file other than **99user.ldif**, edit the file directly. Neither the Directory Server Console nor LDAP tools can edit a schema file other than **99user.ldif**.

8.6. CREATING CUSTOM SCHEMA FILES

Schema files are simple LDIF files which define the **cn=schema** entry. Each attribute and object class is added as an attribute to that entry. Here are the requirements for creating a schema file:

- The first line must be **dn: cn=schema**.
- The schema file can include both attributes and object classes, but it can also include only one or the other.
- If both attributes and object classes are defined in the style, all of the attributes must be listed in the file first, then all of the attributes must be listed first, then the object classes.
- The object classes can use attributes defined in other schema files.
- The file must be named in the format **[1-9][0-9]text.ldif**.

The file must always begin with two numbers. Numerically, the schema file cannot be loaded before the core configuration schema (which begin with **00** and **01**).

Also, the Directory Server always writes its custom schema to the numerically and alphabetically highest named schema file in the schema directory. It expects this file to be **99user.ldif**. If this file is not **99user.ldif**, the server can experience problems. So, always make sure custom schema files are at least alphabetically lower than **99user.ldif**. The name **99alpha.ldif** is okay; the name **99zzz.ldif** is not.

Practices for creating schema files are described in more detail in the *Deployment Guide*.

Attributes are defined in the schema file as ***attributetypes*** attributes to the schema, with five components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The OID for the syntax of the attribute values, listed in Table 8.1, "Supported LDAP Attribute Syntaxes" , in the form **SYNTAX** *OID*
- Optionally, the source where the attribute is defined

For example:

```
attributetypes: ( 1.2.3.4.5.6.1 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUED X-ORIGIN 'Example defined')
```

Likewise, object classes are defined as ***objectclasses*** attributes, although there is slightly more flexibility in how the object class is defined. The only required configurations are the name and OID for the object class; all other configuration depends on the needs for the object class:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The superior, or parent, object class for this object class, in the form **SUP** *object_class*; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry
- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)
- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)

For example:

```
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object Class' SUP
inetOrgPerson AUXILIARY  MUST cn MAY (exampleDateOfBirth $ examplePreferredOS) )
```

Example 8.4, "Example Schema File" shows a simplified schema file.

Example 8.4. Example Schema File

```
dn: cn=schema
attributetypes: ( 2.16.840.1133730.1.123 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'Example defined')
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object Class' SUP
inetOrgPerson AUXILIARY MAY (dateofbirth) )
```

Custom schema files should be added to the Directory Server instance's schema directory, **/etc/dirsrv/slapd-*instance_name*/schema**. The schema in these files are not loaded and available to the server unless the server is restarted or a dynamic reload task is run.

8.7. DYNAMICALLY RELOADING SCHEMA

By default, the schema files used by the Directory Server instance are loaded into the directory when it is started. This means that any new schema files added to the schema directory are not available for use unless the server is restarted. The Directory Server has a task which manually reloads the full schema for the Directory Server instance, including custom files, without requiring a server restart.

The schema reload task can be initiated in two ways:

- Using the **schema-reload.pl** script
- Adding a **cn=schema reload** task entry using **ldapmodify**

8.7.1. Reloading Schema Using schema-reload.pl

The **schema-reload.pl** script launches a special task to reload all of the schema files used by a specific Directory Server instance. This allows custom schema files to be loaded dynamically without having to add schema elements to **99user.ldif**.

1. Open the tool directory for the Directory Server instance, **/usr/lib/dirsrv/slapd-*instance_name*/**.
2. Run the script, binding as the Directory Manager.

   ```
   ./schema-reload.pl -D "cn=Directory Manager" -w secret
   ```

   The Directory Server responds that it has added the new reload task entry.

   ```
   adding new entry cn=schema_reload_2009_1_6_17_52_4,cn=schema reload task,cn=tasks,cn=config
   ```

   This reloads the schema from the default schema directory, **/etc/dirsrv/slapd-*instance_name*/schema**, which is recommended. It is also possible to specify a different directory using the **-d** option.

   ```
   ./schema-reload.pl -D "cn=Directory Manager" -w password -d /export/custom-schema
   ```

IMPORTANT

All of the schema is reloaded with the schema reload task, not just the newest schema or modified schema. This means that whatever directory is specified should contain the full schema for the Directory Server, or the Directory Server instance may have serious performance problems.

The **schema-reload.pl** is described in more detail in the *Configuration and Command-Line Tool Reference* .

8.7.2. Reloading Schema Using ldapmodify

The **schema-reload.pl** script creates a special task entry in a Directory Server instance which reloads schema files; it is also possible to reload schema by creating the task entry directly. Task entries occur under the **cn=tasks** configuration entry in the **dse.ldif** file, so it is also possible to initiate a task by adding the entry using **ldapmodify**. As soon as the task is complete, the entry is removed from the directory.

To initiate a schema reload task, add an entry under the **cn=schema reload task,cn=tasks,cn=config** entry. The only required attribute is the ***cn*** for the specific task.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example schema reload,cn=schema reload task,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn:example schema reload
```

The default schema directory from which the Directory Server instance reloads the schema is in **/etc/dirsrv/slapd-*instance_name*/schema**; it is possible to specify a different schema directory using the ***schemadir*** attribute, which is analogous to the **-d** option with **schema-reload.pl**.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example schema reload,cn=schema reload task,cn=tasks,cn=config
```

```
changetype: add
objectclass: extensibleObject
cn:example schema reload
schemadir: /export/schema
```

IMPORTANT

All of the schema is reloaded with the schema reload task, not just the newest schema or modified schema. This means that whatever directory is specified should contain the full schema for the Directory Server, or the Directory Server instance may have serious performance problems.

As soon as the task is completed, the entry is deleted from the **dse.ldif** configuration, so it is possible to reuse the same task entry continually.

The **cn=schema reload task** configuration is described in more detail in the *Configuration and Command-Line Tool Reference*.

8.7.3. Reloading Schema with Replication

The schema reload task is a local operation, so schema changes are not replicated in a multi-master environment if the schema is added to one supplier but not to the others. To load the new schema files on all of the supplier servers:

1. Stop replication.
2. Copy the new schema file over and run the schema reload task for every supplier and replica server.
3. Restart replication.

8.7.4. Schema Reload Errors

When the schema reload task runs, the command prompt only shows that the task is initiated.

```
adding new entry cn=schema reload task 1,cn=schema reload task,cn=tasks,cn=config
```

However, the task does not return whether it completed successfully. To verify the schema reload operation was successful, check the error logs. The schema reload has two tasks, first validating the schema file and then loading it.

A success message shows that the validation passed and the task finished.

```
[06/Jan/2009:17:52:04 -0500] schemareload - Schema reload task starts (schema dir: default) ...
[06/Jan/2009:17:52:04 -0500] schemareload - Schema validation passed.
[06/Jan/2009:17:52:04 -0500] schemareload - Schema reload task finished.
```

If there is a failure, then the logs show which step failed and why.

```
[..] schemareload - Schema reload task starts (schema dir: /bogus) ...
[..] schema - No schema files were found in the directory /bogus
[..] schema_reload - schema file validation failed
[..] schemareload - Schema validation failed.
```

8.8. TURNING SCHEMA CHECKING ON AND OFF

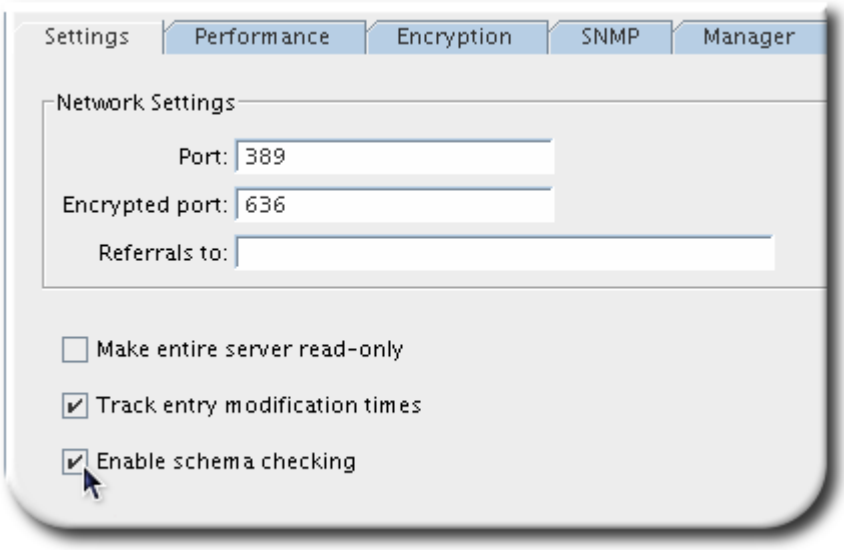
When schema checking is on, the Directory Server ensures three things:

- The object classes and attributes using are defined in the directory schema.
- The attributes required for an object class are contained in the entry.
- Only attributes allowed by the object class are contained in the entry.

Schema checking is turned on by default in the Directory Server, and the Directory Server should always run with schema checking turned on. The only situation where is may be beneficial to turn schema checking off is to accelerate LDAP import operations. However, there is a risk of importing entries that do not conform to the schema. Consequently, it is impossible to search for these entries.

1. In the Directory Server Console, select the **Configuration** tab.

2. Highlight the server icon at the top of the navigation tree, then select the **Settings** tab in the right pane.
3. To enable schema checking, check the **Enable Schema Checking** check box; clear it to turn off schema checking.



4. Click **Save**.

To turn schema checking on and off using LDAP commands, edit the value of the ***nsslapd-schemacheck*** attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-schemacheck: on
nsslapd-schemacheck: off
```

For information, see the *Directory Server Configuration and Command-Line Tool Reference* .

8.9. USING SYNTAX VALIDATION

With syntax validation, the Directory Server checks that the value of an attribute follows the rules of the syntax given in the definition for that attribute. For example, syntax validation will confirm that a new ***telephoneNumber*** attribute actually has a valid telephone number for its value.

8.9.1. About Syntax Validation

As with schema checking, validation reviews any directory modification and rejects changes that violate the syntax. Additional settings can be optionally configured so that syntax validation can log warning messages about syntax violations and then either reject the modification or allow the modification process to succeed.

This feature validates all attribute syntaxes, with the exception of binary syntaxes (which cannot be verified) and non-standard syntaxes, which do not have a defined required format. The syntaxes are validated against RFC 4514.

8.9.2. Syntax Validation and Other Directory Server Operations

Syntax validation is mainly relevant for standard LDAP operations like creating entries (add) or editing attributes (modify). Validating attribute syntax can impact other Directory Server operations, however.

Database Encryption

For normal LDAP operations, an attribute is encrypted just before the value is written to the database. This means That encryption occurs *after* the attribute syntax is validated.

Encrypted databases (as described in Section 2.2.3, "Configuring Attribute Encryption") can be exported and imported. Normally, it is strongly recommended that these export and import operations are done with the **-E** flag with **db2ldif** and **ldif2db**, which allows syntax validation to occur just fine for the import operation. However, if the encrypted database is exported without using the **-E** flag (which is not supported), then an LDIF with encrypted values is created. When this LDIF is then imported, the encrypted attributes cannot be validated, a warning is logged, and attribute validation is skipped in the imported entry.

Synchronization

There may be differences in the allowed or enforced syntaxes for attributes in Windows Active Directory entries and Red Hat Directory Server entries. In that case, the Active Directory values could not be properly synced over because syntax validation enforces the RFC standards in the Directory Server entries.

Replication

If the Directory Server 9.0 instance is a supplier which replicates its changes to a consumer, then there is no issue with using syntax validation. However, if the supplier in replication is an older version of Directory Server or has syntax validation disabled, then syntax validation should not be used on the 9.0 consumer because the Directory Server 9.0 consumer may reject attribute values that the master allows.

8.9.3. Enabling or Disabling Syntax Validation

Syntax validation is configured by the ***nsslapd-syntaxcheck*** attribute. The value of this attribute is either **on** or **off** (by default, this is on). To change the syntax validation, modify this attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-syntaxcheck
nsslapd-syntaxcheck: off
```

NOTE

If syntax validation is disabled, then run the **syntax-validate.pl** script to audit existing attribute values before re-enabling syntax validation. See Section 8.9.6, "Validating the Syntax of Existing Attribute Values".

8.9.4. Enabling Strict Syntax Validation for DNs

When syntax validation is enabled, DNs are validated against RFC 4514, as are other attribute syntaxes. However, DN syntax validation is enabled separately because the strictness of later standards can invalidate old-style DNs, and therefore directory trees.

Syntax validation checks DNs against section 3 in RFC 4514.

The value of this attribute is either **on** or **off** (by default, this is off). To change the syntax validation, modify this attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-dn-validate-strict
nsslapd-dn-validate-strict: on
```

NOTE

If strict DN validation is enabled and a DN value does not conform to the required syntax, then the operation fails with LDAP result code 34, INVALID_DN_SYNTAX.

8.9.5. Enabling Syntax Validation Warnings (Logging)

By default, syntax validation rejects any add or modify operations where an attribute value violates the required syntax. However, the violation itself is not recorded to the errors log by default. The ***nsslapd-syntaxlogging*** attribute enables error logging for any syntax violations.

NOTE

Syntax violations discovered by the syntax validation script and task are logged in the Directory Server error log.

If ***nsslapd-syntaxlogging*** and ***nsslapd-syntaxcheck*** are both enabled, then any invalid attribute modification is rejected and the message written to the log. If ***nsslapd-syntaxlogging*** is enabled but ***nsslapd-syntaxcheck*** is disabled, then the operation is allowed to succeed, but the warning message is still written to the error log.

The value of this attribute is either **on** or **off** (by default, this is off). To enable syntax validation logging, edit the attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-syntaxlogging
nsslapd-syntaxlogging: on
```

8.9.6. Validating the Syntax of Existing Attribute Values

Syntax validation checks every modification to attributes to make sure that the new value has the required syntax for that attribute type. However, syntax validation only audits *changes* to attribute values, such as when an attribute is added or modified. It does not validate the syntax of *existing* attribute values.

Validation of existing attribute values can be done with the syntax validation script. This script checks entries under a specified subtree (in the **-b** option) and, optionally, only entries which match a specified filter (in the **-f** option). For example:

```
/usr/lib64/dirsrv/instance_name/syntax-validate.pl -D "cn=directory manager" -w secret -b
"ou=people,dc=example,dc=com" -f "(objectclass=inetorgperson)"
```

The script identifies syntax violations, however, you must fix them manually.

NOTE

If syntax validation is disabled or if a server is migrated, then there may be data in the server which does not conform to attribute syntax requirements. The syntax validation script can be run to evaluate those existing attribute values before enabling syntax validation.

Alternately, a task can be launched to initiate syntax validation, specifying the required base DN and, optionally, LDAP search filter.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example,cn=syntax validation,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn:example
basedn: ou=people,dc=example,dc=com
filter: "(objectclass=inetorgperson)"
```

CHAPTER 9. MANAGING INDEXES

Indexing makes searching for and retrieving information easier by classifying and organizing attributes or values. This chapter describes the searching algorithm itself, placing indexing mechanisms in context, and then describes how to create, delete, and manage indexes.

9.1. ABOUT INDEXES

This section provides an overview of indexing in Directory Server. It contains the following topics:

9.1.1. About Index Types

Indexes are stored in files in the directory's databases. The names of the files are based on the indexed attribute, not the type of index contained in the file. Each index file may contain multiple types of indexes if multiple indexes are maintained for the specific attribute. For example, all indexes maintained for the common name attribute are contained in the **cn.db4** file.

Directory Server supports the following types of index:

- *Presence index (pres)* contains a list of the entries that contain a particular attribute, which is very useful for searches. For example, it makes it easy to examine any entries that contain access control information. Generating an **aci.db4** file that includes a presence index efficiently performs the search for **ACI=*** to generate the access control list for the server.

  The presence index is not used for base object searches.

- *Equality index (eq)* improves searches for entries containing a specific attribute value. For example, an equality index on the ***cn*** attribute allows a user to perform the search for **cn=Babs Jensen** far more efficiently.

- *Approximate index (approx)* is used for efficient approximate or *sounds-like* searches. For example, an entry may include the attribute value **cn=Robert E Lee**. An approximate search would return this value for searches against **cn~=Robert Lee**, **cn~=Robert**, or **cn~=Lee**. Similarly, a search against **l~=San Fransisco** (note the misspelling) would return entries including **l=San Francisco**.

- *Substring index (sub)* is a costly index to maintain, but it allows efficient searching against substrings within entries. Substring indexes are limited to a minimum of three characters for each entry.

  For example, searches of the form **cn=*derson** , match the common names containing strings such as **Bill Anderson**, **Jill Henderson**, or **Steve Sanderson**. Similarly, the search for **telephoneNumber= *555*** returns all the entries in the directory with telephone numbers that contain **555**.

- *International index* speeds up searches for information in international directories. The process for creating an international index is similar to the process for creating regular indexes, except that it applies a *matching rule* by associating an *object identifier* (OID) with the attributes to be indexed.

  The supported locales and their associated OIDs are listed in Appendix D, *Internationalization*. If there is a need to configure the Directory Server to accept additional matching rules, contact Red Hat Professional Services.

- *Browsing index*, or *virtual list view (VLV) index* , speeds up the display of entries in the Directory Server Console. This index is particularly useful if a branch of your directory contains hundreds of entries; for example, the **ou=people** branch. You can create a browsing index on any branch point in the directory tree to improve display performance through the Directory Server Console or by using the **vlvindex** command-line tool, which is explained in the *Directory Server Configuration and Command-Line Tool Reference* .

9.1.2. About Default, System, and Standard Indexes

When you install Directory Server, a set of default and system indexes is created per database instance. To maintain these indexes, the directory uses standard indexes.

9.1.2.1. Overview of Default Indexes

The default indexes can be modified depending on the directory indexing needs. Always ensure that no server plug-ins or other servers depend on a default index before removing it.

Table 9.1, "Default Indexes" lists the default indexes installed with the directory.

Table 9.1. Default Indexes

Attribute	Eq	Pres	Sub	Purpose
cn	●	●	●	Improves the performance of the most common types of user directory searches.
givenname	●	●	●	Improves the performance of the most common types of user directory searches.
mail	●	●	●	Improves the performance of the most common types of user directory searches.
mailHost	●			Used by a messaging server.
member	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 3.6, "Maintaining Referential Integrity" for more information.
owner	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 3.6, "Maintaining Referential Integrity" for more information.
see Also	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 3.6, "Maintaining Referential Integrity" for more information.
sn	●	●	●	Improves the performance of the most common types of user directory searches.

Attribute	Eq	Pres	Sub	Purpose
telephoneNumber	●	●	●	Improves the performance of the most common types of user directory searches.
uid	●			Improves Directory Server performance.
unique member	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 3.6, "Maintaining Referential Integrity" for more information.

9.1.2.2. Overview of System Indexes

System indexes cannot be deleted or modified. They are required by the directory to function properly. Table 9.2, "System Indexes" lists the system indexes included with the directory.

Table 9.2. System Indexes

Attribute	Eq	Pres	Purpose
aci		●	Allows the Directory Server to quickly obtain the access control information maintained in the database.
objectClass	●		Used to help accelerate subtree searches in the directory.
entryDN	●		Speeds up entry retrieval based on DN searches.
parentID	●		Enhances directory performance during one-level searches.
numSubordinates		●	Used by the Directory Server Console to enhance display performance on the **Directory** tab.
nsUniqueID	●		Used to search for specific entries.

9.1.2.3. Overview of Standard Indexes

Because of the need to maintain default indexes and other internal indexing mechanisms, the Directory Server also maintains certain standard index files. The standard index, **id2entry.db4**, exists by default in Directory Server; you do not need to generate it.

The **id2entry.db4** contains the actual directory database entries. All other database files can be recreated from this one.

9.1.3. Overview of the Searching Algorithm

Indexes are used to speed up searches. To understand how the directory uses indexes, it helps to understand the searching algorithm. Each index contains a list of attributes (such as the ***cn***, common name, attribute) and a pointer to the entries corresponding to each value. Directory Serverprocesses a search request as follows:

1. An LDAP client application sends a search request to the directory.
2. The directory examines the incoming request to make sure that the specified base DN matches a suffix contained by one or more of its databases or database links.
    - If they do match, the directory processes the request.
    - If they do not match, the directory returns an error to the client indicating that the suffix does not match. If a referral has been specified in the **nsslapd-referral** attribute under **cn=config**, the directory also returns the LDAP URL where the client can attempt to pursue the request.
    - The Directory Server examines the search filter to see what indexes apply, and it attempts to load the list of entry IDs from each index that satisfies the filter. The ID lists are combined based on whether the filter used AND or OR joins.
    - If the list of entry IDs is larger than the configured ID list scan limit or if there is no index, then the Directory Server searches every entry in the database. This is an *unindexed* search.
3. The Directory Server reads every entry from the **id2entry.db4** database or the entry cache for every entry ID in the ID list (or from the entire database for an unindexed search). The server then checks the entries to see if they match the search filter. Each match is returned as it is found.

   The server continues through the list of IDs until it has searched all candidate entries or until it hits one of the configured resource limits. (Resource limits are listed in Table 10.1, "Resource Limit Attributes".)

NOTE

It's possible to set separate resource limits for searches using the simple paged results control. For example, administrators can set high or unlimited size and look-through limits with paged searches, but use the lower default limits for non-paged searches.

9.1.4. Approximate Searches

In addition, the directory uses a variation of the metaphone phonetic algorithm to perform searches on an approximate index. Each value is treated as a sequence of words, and a phonetic code is generated for each word.

NOTE

The metaphone phonetic algorithm in Directory Server supports only US-ASCII letters. Therefore, use approximate indexing only with English values.

Values entered on an approximate search are similarly translated into a sequence of phonetic codes. An entry is considered to match a query if both of the following are true:

- All of the query string codes match the codes generated in the entry string.
- All of the query string codes are in the same order as the entry string codes.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
Alice B Sarette (ALS B SRT)	Alice Sarette (ALS SRT)	Matches. Codes are specified in the correct order.
	Alice Sarrette (ALS SRT)	Matches. Codes are specified in the correct order, despite the misspelling of Sarette.
	Surette (SRT)	Matches. The generated code exists in the original name, despite the misspelling of Sarette.
	Bertha Sarette (BR0 SRT)	No match. The code BR0 does not exist in the original name.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
	Sarette, Alice (SRT ALS)	No match. The codes are not specified in the correct order.

9.1.5. Indexing Performance

Each index that the directory uses is composed of a table of index keys and matching entry ID lists. This entry ID list is used by the directory to build a list of candidate entries that may match a client application's search request; Section 9.1, "About Indexes" describes each kind of Directory Server index. The Directory Server secondary index structure greatly improves write and search operations.

While achieving extremely high read performance, in previous versions of Directory Server, write performance was limited by the number of bytes per second that could be written into the storage manager's transaction log file. Large log files were generated for each LDAP write operation; in fact, *log file verbosity* could easily be 100 times the corresponding number of bytes changed in the Directory Server. The majority of the contents in the log files are related to index changes (ID insert and delete operations).

The secondary index structure was separated into two levels in the old design:

- The ID list structures, which were the province of the Directory Server back end and opaque to the storage manager.
- The storage manager structures (Btrees), which were opaque to the Directory Server back end code.

Because it had no insight into the internal structure of the ID lists, the storage manager had to treat ID lists as opaque byte arrays. From the storage manager's perspective, when the content of an ID list changed, the **entire list** had changed. For a single ID that was inserted or deleted from an ID list, the corresponding number of bytes written to the transaction log was the maximum configured size for that ID list, about 8 kilobytes. Also, every database page on which the list was stored was marked as dirty, since the *entire* list had changed.

In the redesigned index, the storage manager has visibility into the fine-grain index structure, which optimizes transaction logging so that only the number of bytes actually changed need to be logged for any given index modification. The Berkeley DB provides ID list semantics, which are implemented by the storage manager. The Berkeley API was enhanced to support the insertion and deletion of individual IDs stored against a common key, with support for duplicate keys, and an optimized mechanism for the retrieval of the complete ID list for a given key.

The storage manager has direct knowledge of the application's intent when changes are made to ID lists, resulting in several improvements to ID list handling:

- For long ID lists, the number of bytes written to the transaction log for any update to the list is significantly reduced, from the maximum ID list size (8 kilobytes) to twice the size of one ID (4 bytes).
- For short ID lists, storage efficiency, and in most cases performance, is improved because only the storage manager meditate need to be stored, not the ID list metadata.
- The average number of database pages marked as dirty per ID insert or delete operation is very small because a large number of duplicate keys will fit into each database page.

9.1.6. Balancing the Benefits of Indexing

Before creating new indexes, balance the benefits of maintaining indexes against the costs.

- Approximate indexes are not efficient for attributes commonly containing numbers, such as telephone numbers.
- Substring indexes do not work for binary attributes.
- Equality indexes should be avoided if the value is big (such as attributes intended to contain photographs or passwords containing encrypted data).
- Maintaining indexes for attributes not commonly used in a search increases overhead without improving global searching performance.
- Attributes that are not indexed can still be specified in search requests, although the search performance may be degraded significantly, depending on the type of search.
- The more indexes you maintain, the more disk space you require.

Indexes can become very time-consuming. For example:

1. The Directory Server receives an add or modify operation.
2. The Directory Server examines the indexing attributes to determine whether an index is maintained for the attribute values.
3. If the created attribute values are indexed, then the Directory Server generates the new index entries.
4. Once the server completes the indexing, the actual attribute values are created according to the client request.

For example, the Directory Server adds the entry:

```
dn: cn=John Doe,ou=People,dc=example,dc=com
objectclass: top
objectClass: person
objectClass: orgperson
objectClass: inetorgperson
cn: John Doe
cn: John
sn: Doe
ou: Manufacturing
ou: people
telephoneNumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.
```

The Directory Server maintains the following indexes:

- Equality, approximate, and substring indexes for ***cn*** (common name) and ***sn*** (surname) attributes.
- Equality and substring indexes for the telephone number attribute.
- Substring indexes for the description attribute.

When adding that entry to the directory, the Directory Server must perform these steps:

1. Create the ***cn*** equality index entry for **John** and **John Doe**.
2. Create the appropriate ***cn*** approximate index entries for **John** and **John Doe**.
3. Create the appropriate ***cn*** substring index entries for **John** and **John Doe**.
4. Create the ***sn*** equality index entry for **Doe**.
5. Create the appropriate ***sn*** approximate index entry for **Doe**.
6. Create the appropriate ***sn*** substring index entries for **Doe**.
7. Create the telephone number equality index entry for **408 555 8834**.
8. Create the appropriate telephone number substring index entries for **408 555 8834**.
9. Create the appropriate description substring index entries for **Manufacturing lead for the Z238 line of widgets**. A large number of substring entries are generated for this string.

As this example shows, the number of actions required to create and maintain databases for a large directory can be resource-intensive.

9.2. CREATING STANDARD INDEXES

This section describes how to create presence, equality, approximate, substring, and international indexes for specific attributes using the Directory Server Console and the command line.

When a new index type is created, that index is used as a template for any additional databases as they are added. The Directory Server uses the current set of default indexes defined for the instance as the basis for additional databases.

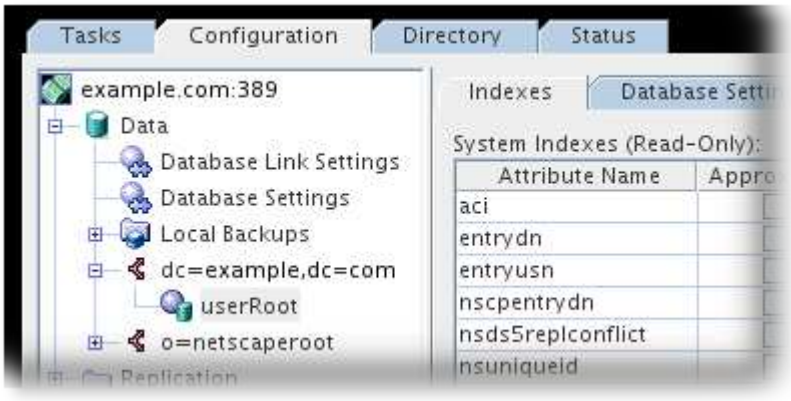
However, new indexes are not added to existing databases automatically, though they can be added manually. This means that if you add a default index to your second database instance, it will not be maintained in your first database instance but will be maintained in any subsequent instances. To apply a new index to an existing database, run the **db2index.pl** script or run a **cn=index,cn=tasks** task, as described in Section 9.3, “Applying New Indexes to Existing Databases”.

- Section 9.2.1, “Creating Indexes from the Server Console”
- Section 9.2.2, “Creating Indexes from the Command Line”

9.2.1. Creating Indexes from the Server Console

To create presence, equality, approximate, substring, or international indexes:

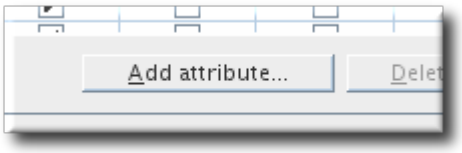
1. Select the **Configuration** tab.
2. Expand the **Data** node, expand the suffix of the database to index, and select the database.
3. Select the **Indexes** tab in the right pane.



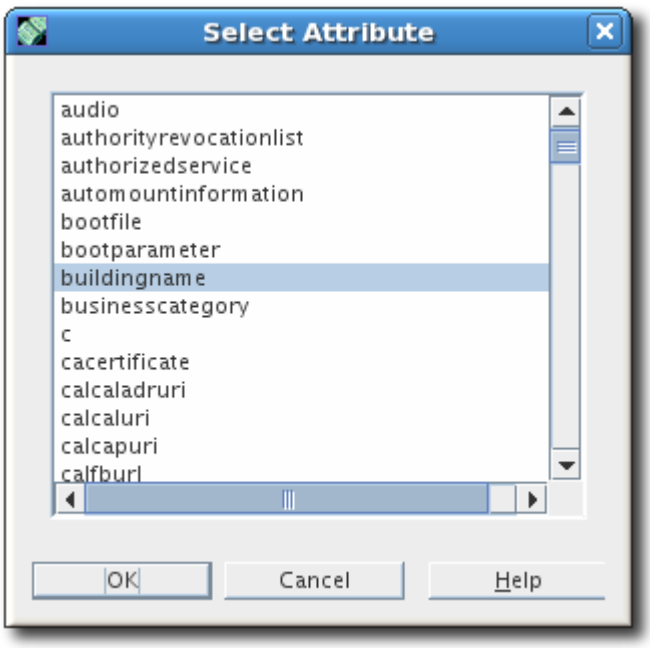
NOTE

Do not click the **Database Settings** node because this opens the **Default Index Settings** window, not the window for configuring indexes per database.

4. If the attribute to be indexed is listed in the **Additional Indexes** table, go to step 6. Otherwise, click **Add Attribute** to open a dialog box with a list of all of the available attributes in the server schema.

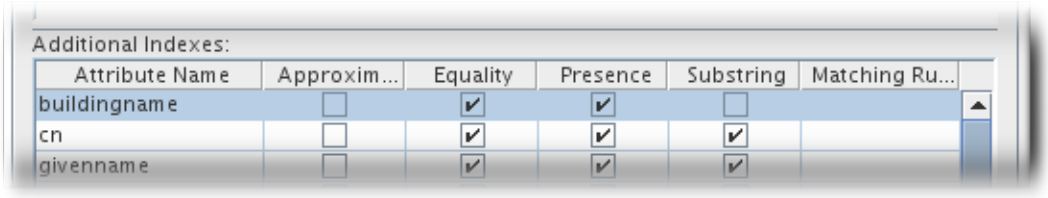


5. Select the attribute to index, and click **OK**.



The server adds the attribute to the **Additional Indexes** table.

6. Select the check box for each type of index to maintain for each attribute.



7. To create an index for a language other than English, enter the OID of the *collation order* to use in the **Matching Rules** field.

   To index the attribute using multiple languages, list multiple OIDs separated by commas, but no whitespace. For a list of languages, their associated OIDs, and further information regarding collation orders, see Appendix D, *Internationalization*.

8. Click **Save**.

The new index is immediately active for any new data that you add and any existing data in your directory. You do not have to restart your server.

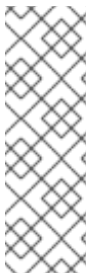
9.2.2. Creating Indexes from the Command Line

NOTE

You cannot create new system indexes because system indexes are hard-coded in Directory Server.

Use **ldapmodify** to add the new index attributes to your directory.

- To create a new index that will become one of the default indexes, add the new index attributes to the **cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config** entry.
- To create a new index for a particular database, add it to the **cn=index,cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** entry, where **cn=***database_name* corresponds to the name of the database.



NOTE

Avoid creating entries under **cn=config** in the **dse.ldif** file. The **cn=config** entry in the simple, flat **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will probably suffer. Although we recommend you do not store simple user entries under **cn=config** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **cn=config** since this centralizes configuration information.

For information on the LDIF update statements required to add entries, see Section 3.3, "Using LDIF Update Statements to Create or Modify Entries".

For example, to create presence, equality, and substring indexes for the ***sn*** (surname) attribute in the **Example1** database:

1. Run **ldapmodify** and add the LDIF entry for the new indexes:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
```

```
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

The ***cn*** attribute contains the name of the attribute to index, in this example the ***sn*** attribute. The entry is a member of the **nsIndex** object class. The **nsSystemIndex** attribute is **false**, indicating that the index is not essential to Directory Server operations. The multi-valued **nsIndexType** attribute specifies the presence (**pres**), equality (**eq**) and substring (**sub**) indexes. Each keyword has to be entered on a separate line. The **nsMatchingRule** attribute in the example specifies the OID of the Bulgarian collation order; the matching rule can indicate any possible value match, such as languages or other formats like date or integer.

You can use the keyword **none** in the **nsIndexType** attribute to specify that no indexes are to be maintained for the attribute. This example temporarily disables the ***sn*** indexes on the **Example1** database by changing the **nsIndexType** to **none**:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:none
```

For a complete list of matching rules and their OIDs, see Section 10.4.4, “Using Matching Rules”, and for the index configuration attributes, see the *Directory Server Configuration and Command-Line Tool Reference* .

NOTE

Always use the attribute's primary name (not the attribute's alias) when creating indexes. The primary name of the attribute is the first name listed for the attribute in the schema; for example, **uid** for the user ID attribute.

9.3. APPLYING NEW INDEXES TO EXISTING DATABASES

New indexes are not added to existing databases automatically. They must be added manually, and Directory Server has two methods for applying new indexes to an existing database: running the **db2index.pl** script or running a **cn=index,cn=tasks** task.

9.3.1. Running the db2index.pl Script

After creating an indexing entry or adding additional index types to an existing indexing entry, run the **db2index.pl** script to generate the new set of indexes to be maintained by the Directory Server. After the script is run, the new set of indexes is active for any new data added to the directory and any existing data in the directory.

Run the **db2index.pl** Perl script.

```
[root@server ~]# /usr/lib[64]/dirsrv/slapd-example/db2index.pl -D "cn=Directory Manager" -w secret -n
ExampleServer -t sn
```

For more information about using this Perl script, see the *Directory Server Configuration and Command-Line Tool Reference*.

Table 9.3, “db2index.pl Options” describes the **db2index.pl** options.

Table 9.3. db2index.pl Options

Option	Description
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
-n	Specifies the name of the database being indexed.
-t	Specifies the name of the attribute contained by the index.

9.3.2. Using a cn=tasks Entry to Create an Index

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=index,cn=tasks,cn=config** to initiate an indexing operation.

This task entry requires a unique name (***cn***) and a definition for the attribute and type of index, set in ***nsIndexAttribute*** in the format *attribute:index_type*.

For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example presence index,cn=index,cn=tasks,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: example presence index
nsInstance: userRoot
nsIndexAttribute: "cn:pres"
```

There are three possible *index_type*s:

- **pres** for presence indexes
- **eq** for equality indexes
- **sub** for substring indexes

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on running Directory Server tasks under the **cn=tasks** entries.

9.4. CREATING BROWSING (VLV) INDEXES

A *virtual list view (VLV) index* is a way of creating a truncated list for faster searching while enhancing server performance. The VLV index itself can be resource-intensive to maintain, but it can be beneficial in large directories (over 1000 entries).

A browsing index is a type of VLV index that organizes the entries listed into alphabetical order, making it easier to find entries.

VLV indexes are not applied to attributes, like standard indexes are, but they are dynamically generated based on attributes set in entries and the location of those entries in the directory tree. VLV indexes, unlike standard indexes, are special entries in the database rather than configuration settings for the database.

NOTE

VLV indexes are similar to simple paged results, which can be returned with some external LDAP clients. Simple paged results are calculated per search, while VLV indexes are a permanent list, so VLV indexes are overall faster for searches, but do require some overhead for the server to maintain.

Simple paged results and VLV indexes *cannot* be used on the same search.

For more information, see Section 10.7.4, “Using Simple Paged Results” .

9.4.1. Creating Browsing Indexes from the Server Console

1. Select the **Directory** tab.
2. In the left navigation tree, select the entry, such as **People**, for which to create the index.
3. From the **Object** menu, select **Create Browsing Index**.

The **Create Browsing Index** dialog box appears displaying the status of the index creation. Click the **Status Logs** box to view the status of the indexes created.

4. Click **Close**.

The new index is immediately active for any new data that is added to the directory. You do not have to restart your server.

For more information on how to change the VLV search information or the access control rules that are set by default for VLV searches, see Section 9.4.2.1, "Adding a Browsing Index Entry" and Section 9.4.3, "Setting Access Control for VLV Information".

9.4.2. Creating Browsing Indexes from the Command Line

Creating a browsing index or virtual list view (VLV) index from the command line has these steps:

1. Using **ldapmodify** to add new browsing index entries or edit existing browsing index entries. See Section 9.4.2.1, "Adding a Browsing Index Entry" .
2. Running the **vlvindex** script to generate the new set of browsing indexes to be maintained by the server. See Section 9.4.2.2, "Running the vlvindex Script". Alternatively, launch an appropriate task under **cn=tasks,cn=config** (Section 9.4.2.3, "Using a cn=tasks Entry to Create a Browsing Index").
3. Ensuring that access control on VLV index information is set appropriately. See Section 9.4.3, "Setting Access Control for VLV Information".

9.4.2.1. Adding a Browsing Index Entry

The type of browsing index entry to create depends on the type of **ldapsearch** attribute sorting to accelerate. It is important to take the following into account:

- The scope of the search (base, one, sub)
- The base of the search (the entry to use as a starting point for the search)
- The attributes to sort
- The filter of the search

  For more information on specifying filters for searches, see Chapter 10, *Finding Directory Entries*.

- The LDBM database to which the entry that forms the base of the search belongs. You can only create browsing indexes in LDBM databases.

For example, create a browsing index to accelerate an **ldapsearch** on the entry **ou=People,dc=example,dc=com** held in the **Example1** database with the following attributes:

- The search base is **ou=People,dc=example,dc=com**
- The search filter is **(|(objectclass=*)(objectclass=ldapsubentry))**
- The scope is **one**
- The sorting order for the returned attributes is ***cn***, ***givenname***, ***o***, ***ou***, and ***sn***

1. Run **ldapmodify** and add an entry which specifies the base, scope, and filter of the browsing index:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People,dc=example,dc=com
vlvScope: 1
vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))
```

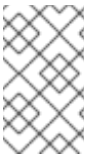
   - The ***cn*** contains the browsing index identifier, which specifies the entry on which to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry. Red Hat recommends using the **dn** of the entry for the browsing index identifier, which is the approach adopted by the Directory Server Console, to prevent identical browsing indexes from being created. The entry is a member of the **vlvSearch** object class.
   - The **vlvbase** attribute value specifies the entry on which you want to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry (the browsing index identifier).
   - The **vlvScope** attribute is **1**, indicating that the scope for the search you want to accelerate is **1**. A search scope of **1** means that only the immediate children of the entry specified in the ***cn*** attribute, and not the entry itself, will be searched.
   - The **vlvFilter** specifies the filter to be used for the search; in this example, **(|(objectclass=*)(objectclass=ldapsubentry))**.

2. Add the second entry, to specify the sorting order for the returned attributes:

```
dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
  dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,
  cn= config
objectClass: top
objectClass: vlvIndex
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenName o ou sn
```

   - The ***cn*** contains the browsing index sort identifier. The above ***cn*** is the type created by the Console by default, which has the sorting order as being set *by* the browsing index base. The entry is a member of the **vlvIndex** object class.

- The ***vlvSort*** attribute value specifies the order in which you want your attributes to be sorted; in this example, ***cn***, ***givenName***, ***o***, ***ou***, and then ***sn***.



NOTE

This first browsing index entry must be added to the **cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** directory tree node, and the second entry must be a child of the first entry.

9.4.2.2. Running the vlvindex Script

After creating the two browsing indexing entries or added additional attribute types to an existing indexing browsing entries, run the **vlvindex** script to generate the new set of browsing indexes to be maintained by the Directory Server. After running the script, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

To run the **vlvindex** script:

1. Stop the server.

```
[root@server ~]# service dirsrv stop instance
```

2. Run the **vlvindex** script.

```
[root@server ~]# /etc/dirsrv/slapd-instance_name/vlvindex -n Example1 -T "by MCC ou=people
dc=example dc=com"
```

For more information about using this script, see the *Directory Server Configuration and Command-Line Tool Reference*.

3. Restart the server.

```
[root@server ~]# service dirsrv start instance
```

Table 9.4, “vlvindex Options” describes the **vlvindex** options used in the examples:

Table 9.4. vlvindex Options

Option	Description
-n	Name of the database containing the entries to index.
-T	Browsing index identifier to use to create browsing indexes.

9.4.2.3. Using a cn=tasks Entry to Create a Browsing Index

As an alternative to running the **vlvindex** script, it is possible to initiate an indexing task directly.

NOTE

Running the indexing task is the same as running the **vlvindex** script.

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=index,cn=tasks,cn=config** to initiate an indexing operation.

This task entry requires a unique name (***cn***) and one other attribute, ***nsIndexVLVAttribute***, which gives the name of the browsing index definition entry to use to generate the VLV index.

For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example VLV index,cn=index,cn=tasks,cn=config
changetype: add
```

```
objectclass: extensibleObject
cn: example VLV index
nsIndexVLVAttribute: "by MCC ou=people,dc=example,dc=com"
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on running Directory Server tasks under the **cn=tasks** entries.

9.4.3. Setting Access Control for VLV Information

The default access control instruction (ACI) allows only authenticated users to use the VLV index information. If you additionally require to allow non-authenticated users to use the VLV index information, update the ***aci*** attribute to set the ***userdn*** parameter to **ldap://anyone**:

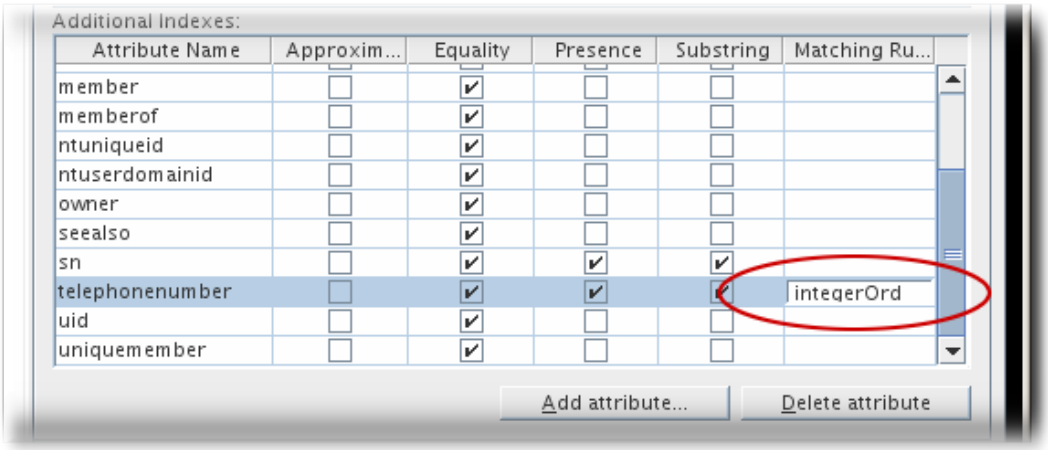
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: aci: (targetattr != "aci")(version 3.0; acl "VLV Request Control";
    allow( read, search, compare, proxy ) userdn = "ldap://anyone" ;)
```

9.5. CHANGING THE INDEX SORT ORDER

By default, indexes are sorted alphabetically, in descending ASCII order. This is true for every attribute, even attributes which may have numeric attribute values like Integer or TelephoneNumber. It is possible to change the sort method by changing the matching rule set for the attribute.

9.5.1. Changing the Sort Order in the Console

1. Select the **Configuration** tab.
2. Expand the **Data** node, expand the suffix of the database to index, and select the database.
3. Select the **Indexes** tab in the right pane.
4. Select the index, and, in the **Matching Rules** field, enter the new sort order to use. For example, to sort by numbers, rather than alphabetically, enter **integerOrderingMatch**.



5. Click **Save**.

9.5.2. Changing the Sort Order in the Command Line

To change the sort order using the command line, change the ***nsMatchingRule*** for the attribute index. For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
```

```
changetype:modify
replace:nsMatchingRule
nsMatchingRule:integerOrderingMatch
```

9.6. CHANGING THE WIDTH FOR INDEXED SUBSTRING SEARCHES

By default, for a search to be indexed, the search string must be at least three characters long, without counting any wildcard characters. For example, the string **abc** would be an indexed search while **ab*** would not be. Indexed searches are significantly faster than unindexed searches, so changing the minimum length of the search key is helpful to increase the number of indexed searches.

To improve search performance, particularly for sites with many wildcard searches, the search string length for indexed searches can be changed. Directory Server has three attributes which allow you to change the minimum number of characters required for an indexed search:

- The ***nsSubStrBegin*** attribute sets the required number of characters for an indexed search for the beginning of a search string, before the wildcard.

  ```
  abc*
  ```

- The ***nsSubStrMiddle*** attribute sets the required number of characters for an indexed search where a wildcard is used in the middle of a search string. For example:

  ```
  ab*z
  ```

- The ***nsSubStrEnd*** attribute sets the required number of characters for an indexed search for the end of a search string, after the wildcard. For example:

  ```
  *xyz
  ```

The default substring search length for the string triplet (before, middle, and end) is 3, 3, and 3, meaning every search requires a minimum of three characters, in every wildcard position.

For any attribute index to have alternate string lengths, add the **extensibleObject** object class to the entry and then set the substring search lengths.

1. Set the new key length for the specific attribute index. This requires adding the **extensibleObject** object class and then adding the ***nsSubStrBegin***, ***nsSubStrEnd***, or ***nsSubStrMiddle*** attributes as appropriate. For example:

   ```
   # ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

   dn: attribute_name,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
   changetype: modify
   add: objectclass
   objectclass: extensibleObject
   -
   add: nsSubStrBegin
   nsSubStrBegin: 2
   -
   add: nsSubStrMiddle
   nsSubStrMiddle: 2
   -
   add: nsSubStrEnd
   nsSubStrEnd: 2
   ```

2. Stop the server.

   ```
   service dirsrv stop
   ```

3. Recreate the attribute index. If even one of the substring search width options is changed, then the entire index must be recreated.

   ```
   db2index -t attribute_name
   ```

4. Start the server again.

```
service dirsrv start
```

9.7. DELETING INDEXES

This section describes how to remove attributes and index types from the index.

9.7.1. Deleting an Attribute from the Default Index Entry

When using the default settings of Directory Server, several attributes listed in the default index entry, such as ***sn***, are indexed. The following attributes are part of the default index:

Table 9.5. Default Index Attributes

aci	cn	entryusn
givenName	mail	mailAlternateAddress
mailHost	member	memberOf
nsUniqueId	ntUniqueId	ntUserDomainId
numsubordinates	objectclass	owner
parentid	seeAlso	sn
telephoneNumber	uid	uniquemember

WARNING

Removing system indexes can significantly affect the Directory Server performance.

For example, to remove the ***sn*** attribute from the default index:

1. Remove the attribute from the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** entry:

   ```
   # ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
   cn=sn,cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config
   ```

   If you do not remove the attribute from this entry, the index for the ***sn*** attribute is automatically recreated and corrupted after the server is restarted.

2. Remove the ***cn=attribute_name,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config*** entry. For details, see:

   - Section 9.7.2, “Removing an Attribute from the Index Using the Server Console”
   - Section 9.7.3, “Removing an Attribute from the Index Using the Command Line”

3. Run the **db2index.pl** Perl script to recreate the index:

   ```
   # db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
   ```

   For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

9.7.2. Removing an Attribute from the Index Using the Server Console

The Directory Server Console can delete any custom indexes, indexes used by other server applications such as a messaging or web server, and default indexes.

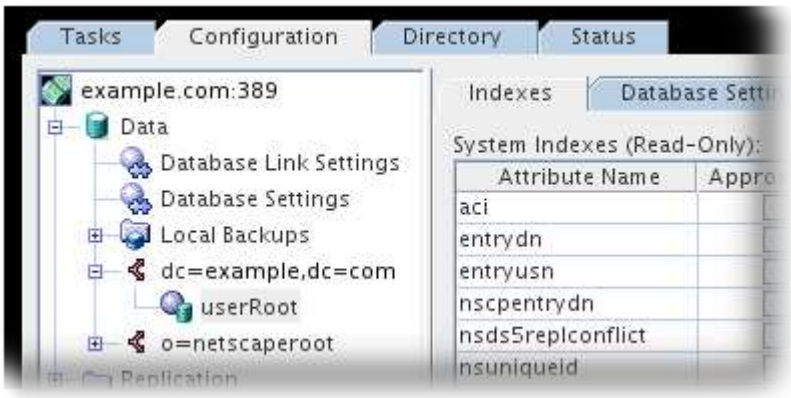
To remove an attribute from the index:

1. If the attribute to remove is listed in the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** default index entry, remove it from this entry first. For details, see Section 9.7.1, "Deleting an Attribute from the Default Index Entry" .

2. Select the **Configuration** tab.

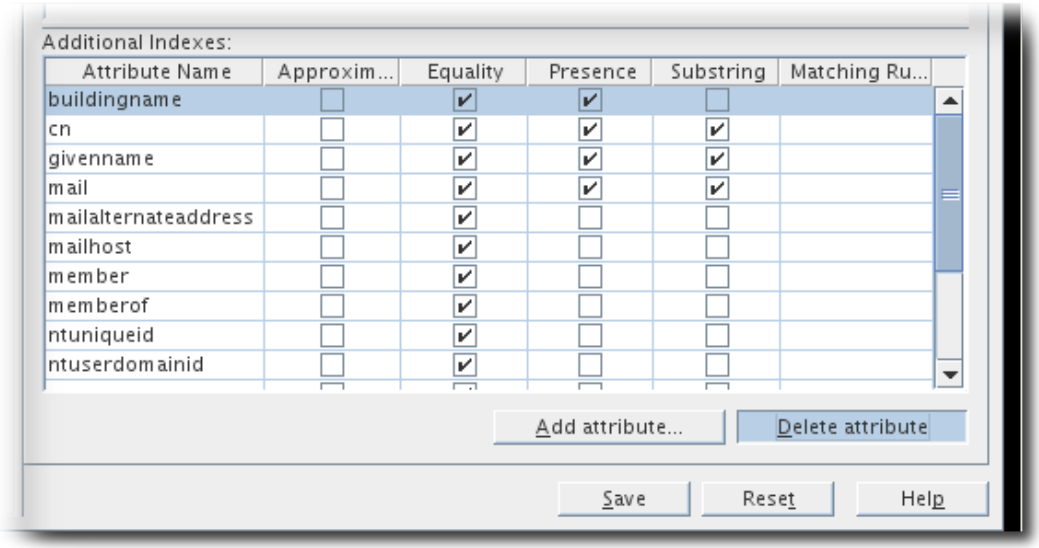
3. Expand the **Data** node and expand the suffix associated with the database containing the index.

4. Select the database from which to delete the index.



5. Locate the attribute containing the index to delete. Clear the check box under the index.

   To delete all indexes maintained for a particular attribute, select the attribute's cell under **Attribute Name**, and click **Delete Attribute**.



6. Click **Save**.

   A **Delete Index** warning dialog box opens, requiring a confirmation to delete the index.

7. Click **Yes** to delete the index.

9.7.3. Removing an Attribute from the Index Using the Command Line

In certain situations you want to remove an attribute from the index. For example, to remove the ***sn*** attribute:

1. If the attribute to remove is listed in the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** default index entry, you must remove it from this entry first. For details, see Section 9.7.1, "Deleting an Attribute from the Default Index Entry" .

2. Remove the attribute from the index:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
```

After deleting the entry, the index for the ***sn*** attribute is no longer maintained.

3. Run the **db2index.pl** Perl script to recreate the index.

```
# db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
```

For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

9.7.4. Deleting Index Types from the Command Line

For example, to remove the **sub** index type of the ***sn*** attribute from the index:

1. Remove the index type:

```
# ldapmodify -D "cn=directory manager" -W -x
dn: cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config

changetype: modify
delete: nsIndexType
nsIndexType:sub
```

After deleting the index entry, the substring index for the ***sn*** attribute is no longer maintained.

2. Run the **db2index.pl** Perl script to recreate the index. For example:

```
# db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
```

For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

9.7.5. Deleting Browsing Indexes from the Server Console

1. Select the **Directory** tab.

2. Select the entry from which to delete the index in the navigation tree, and select **Delete Browsing Index** from the **Object** menu.

Alternatively, select and right-click the entry of the index to delete in the navigation tree, and then choose **Delete Browsing Index** from the pop-up menu.

3. A **Delete Browsing Index** dialog box appears asking you to confirm the deletion of the index. Click **Yes**.

4. The **Delete Browsing Index** dialog box appears displaying the status of the index deletion.

9.7.6. Deleting Browsing Indexes from the Command Line

Deleting a browsing index or virtual list view (VLV) index from the command line involves two steps:

1. Using the **ldapdelete** to delete browsing index entries or edit existing browsing index entries (Section 9.7.6.1, "Deleting a Browsing Index Entry").
2. Running the **vlvindex** script to generate the new set of browsing indexes to be maintained by the server (Section 9.7.6.2, "Running the vlvindex Script"). Alternatively, launch an appropriate task under **cn=tasks,cn=config** (Section 9.4.2.3, "Using a cn=tasks Entry to Create a Browsing Index").

The actual entries for an alphabetical browsing index and virtual list view are the same. The following sections describe the steps involved in deleting browsing indexes.

9.7.6.1. Deleting a Browsing Index Entry

Use the **ldapdelete** command-line utility to either delete browsing indexing entries or edit existing browsing index entries. To delete browsing indexes for a particular database, remove the browsing index entries from the **cn=index,cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** entry, where **cn=***database_name* corresponds to the name of the database.

For example, there is a browsing index for accelerating **ldapsearch** operations on the entry **ou=People,dc=example,dc=com**. It held in the **Example1** database where the search base is **ou=People,dc=example,dc=com**, the search filter is **(|(objectclass=*)(objectclass=ldapsubentry))**, the scope is **1**, and the sorting order for the returned attributes is ***cn***, ***givenname***, ***o***, ***ou***, and ***sn***.

To delete this browsing index, delete the two corresponding browsing index entries:

```
dn: cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People,dc=example,dc=com
vlvScope: 1 vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))

dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: vlvIndex
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenname o ou sn
```

Run **ldapdelete**, specifying both entries.

```
ldapdelete -D "cn=directory manager" -w secret -p 389 -h server.example.com -x "cn=MCC ou=People
dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config" "cn=by MCC ou=People dc=example
dc=com,cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config"
```

After deleting the two browsing index entries, the browsing index will no longer be maintained by the **Example1** database.

9.7.6.2. Running the vlvindex Script

After deleting browsing indexing entries or unwanted attribute types from existing browsing indexing entries, run the **vlvindex** script to generate the new set of browsing indexes to be maintained by the Directory Server. After the script is run, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

1. Stop the server.

   ```
   [root@server ~]# service dirsrv stop instance
   ```

2. Run the **vlvindex** script.

   ```
   [root@server ~]# /etc/dirsrv/slapd-instance_name/vlvindex -n Example1 -T "by MCC ou=people
   dc=example dc=com"
   ```

   For more information about using the **vlvindex** script, see the *Directory Server Configuration and Command-Line Tool Reference*.

3. Restart the server.

```
[root@server ~]# service dirsrv start instance
```

Table 9.4, "vlvindex Options" describes the **vlvindex** options.

Alternatively, create a new task entry under **cn=index,cn=tasks,cn=config** to initiate an indexing operation. This task entry requires a unique name (***cn***) and one other attribute, ***nsIndexVLVAttribute***, which gives the name of the browsing index definition entry to use to generate the VLV index. This task is the same as running **vlvindex**.

For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=example VLV index,cn=index,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example VLV index
nsIndexVLVAttribute: "by MCC ou=people,dc=example,dc=com"
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Directory Server Configuration and Command-Line Tool Reference* has more information on running Directory Server tasks under the **cn=tasks** entries.

CHAPTER 10. FINDING DIRECTORY ENTRIES

Entries in the directory can be searched for and found using any LDAP client. Most clients provide some form of search interface so that the directory can be searched easily and entry information can be easily retrieved.

NOTE

Users cannot search the directory unless the appropriate access control has been set in the directory. For information on setting access control in the directory, see Chapter 13, *Managing Access Control*.

10.1. IMPROVING SEARCH PERFORMANCE THROUGH RESOURCE LIMITS

With large directories, searching through every entry in the database for a search can seriously degrade overall server performance. While this can be alleviated somewhat through effective indexing, indexing itself introduces a performance overhead, and, in large databases, may still not reduce the search scope enough to improve performance.

Reasonable limits can be set on user and client accounts to reduce the total number of entries or the total amount of time spent in an individual search, which both makes searches more responsive and improves overall server performance.

Server limits for search operations are controlled using special operational attribute values on the client application binding to the directory. You can set the following search operation limits:

- *Look through limit.* Specifies how many entries can be examined for a search operation.
- *Size limit.* Specifies the maximum number of entries the server returns to a client application in response to a search operation.
- *Time limit.* Specifies the maximum time the server spends processing a search operation.
- *Idle timeout.* Specifies the time a connection to the server can be idle before the connection is dropped.
- *Range timeout.* Specifies a separate look-through limit specifically for searches using a range.

The resource limits set for the client application take precedence over the default resource limits set for in the global server configuration.

NOTE

The Directory Manager receives unlimited resources by default, with the exception of range searches.

10.1.1. Search Performance and Resource Limits

For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.

10.1.2. Fine Grained ID List Size

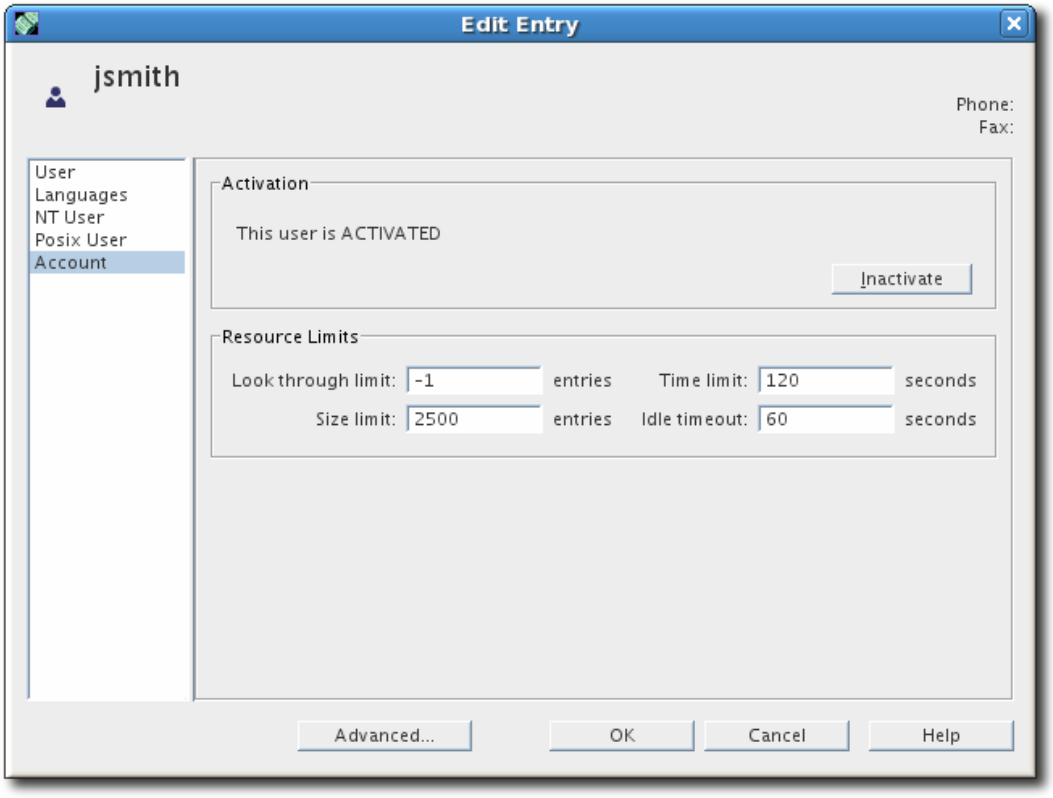
For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.

10.1.3. Setting Resource Limits on a Single User

1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role for which to set resource limits.

   The **Edit Entry** dialog box appears.
3. Click **Account** in the left pane.
4. Set the resource limits. There are four different limits that can be set:
   - *Look through limit*. The maximum number of entries are examined for a search operation.
   - *Size limit*. The maximum number of entries the server returns to a client application in response to a search operation.
   - *Time limit*. The maximum time the server spends processing a search operation.

- *Idle timeout*. The time a connection to the server can be idle before the connection is dropped.



Entering a value of **-1** indicates no limit.

5. Click **OK**.

10.1.4. Setting User and Global Resource Limits Using the Command Line

More options are available when setting resource limits in the command line than through the Directory Server Console. The Directory Server Console sets user-level resource limits. Through the command line, administrators can set user-level resource limits, global resource limits, and limits for specific kinds of searches, such as simple paged and range searches. Section 9.1.3, "Overview of the Searching Algorithm" has more information on how these resource limits affect Directory Server search performance.

Table 10.1, "Resource Limit Attributes" lists operational attributes which can be set for each entry using the command line. Use **ldapmodify** to add the attributes to the entry.

User-level attributes are set on the individual entries, while global configuration attributes are set in the appropriate server configuration area.

Table 10.1. Resource Limit Attributes

User-Level Attribute	Global Configuration Attribute	Global Configuration Entry	Description
nsLookThroughLimit	nsslapd-lookthroughlimit	cn=config,cn=ldbm database,cn=plugins,cn=config	Specifies how many entries are examined for a search operation. Giving this attribute a value of **-1** indicates that there is no limit.
nsPagedLookThroughLimit	nsslapd-pagedlookthroughlimit	cn=config,cn=ldbm database,cn=plugins,cn=config	As with the look-through limit, specifies how many entries are examined, but specifically for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.

User-Level Attribute	Global Configuration Attribute	Global Configuration Entry	Description
nsSizeLimit	nsslapd-sizelimit	cn=config	Specifies the maximum number of entries the server returns to a client application in response to a search operation. Giving this attribute a value of **-1** indicates that there is no limit.
nsPagedSizeLimit	nsslapd-pagedsizelimit	cn=config	As with the size limit, specifies the maximum number of entries the server returns to a client application but only for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.
nsTimeLimit	nsslapd-timelimit	cn=config	Specifies the maximum time the server spends processing a search operation. Giving this attribute a value of **-1** indicates that there is no time limit.
nsidletimeout	nsslapd-idletimeout	cn=config	Specifies the time a connection to the server can be idle before the connection is dropped. The value is given in seconds. Giving this attribute a value of **-1** indicates that there is no limit.
nsIDListScanLimit	nsslapd-idlistscanlimit	cn=config,cn=ldbm database,cn=plugins,cn=config	Specifies the maximum number of entry IDs loaded from an index file for search results. If the ID list size is greater than this value, the search will not use the index list but will treat the search as an unindexed search and look through the entire database.
nsPagedIDListScanLimit	nsslapd-pagedidlistscanlimit	cn=config,cn=ldbm database,cn=plugins,cn=config	As with the ID list scan limit, specifies the maximum number of entry IDs loaded from an index file for search results, but specifically for paged search operations.

User-Level Attribute	Global Configuration Attribute	Global Configuration Entry	Description
	nsslapd-rangelookthroughlimit	cn=config,cn=ldbm database,cn=plugins,cn=config	Specifies how many entries are examined for a range search operation (a search using greater-than, equal-to-or-greater-than, less-than, or equal-to-less-than operators). Giving this attribute a value of **-1** indicates that there is no limit.

For example, this sets the size limit for Barbara Jensen by using **ldapmodify** to modify her entry:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: uid=bjensen,ou=people,dc=example,dc=com
changetype: modify
add: nsSizeLimit
nsSizeLimit: 500
```

The **ldapmodify** statement adds the ***nsSizeLimit*** attribute to Babs Jensen's entry and gives it a search return size limit of 500 entries.

10.1.5. Setting Resource Limits on Anonymous Binds

Resource limits are set on a user entry. An anonymous bind, obviously, does not have a user entry associated with it. This means that the global resource limits usually apply to anonymous operations. However, it is possible to configure resource limits specifically for anonymous binds by creating a template user entry that has resource limits, and then applying that template to anonymous binds.

1. Create a template entry and set whatever resource limits you want to apply to anonymous binds.

   NOTE

   For performance reasons, the template should be in the normal back end, not in the **cn=config** suffix, which does not use an entry cache.

   For example:

   ```
   ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

   dn: cn=anon template,ou=people,dc=example,dc=com
   changetype: add
   objectclass: person
   objectclass: top
   cn: anon template
   sn: template
   nsSizeLimit: 250
   nsLookThroughLimit: 1000
   nsTimeLimit: 60
   ```

2. Add the ***nsslapd-anonlimitsdn*** to the server configuration, pointing to the DN of the template entry. Any of the resource limits in Section 10.1.4, “Setting User and Global Resource Limits Using the Command Line” can be set. For example:

   ```
   [root@server ~]# ldapmodify -D "cn=directory manager" -W -x

   dn: cn=config
   changetype: modify
   add: nsslapd-anonlimitsdn
   nsslapd-anonlimitsdn: cn=anon template,ou=people,dc=example,dc=com
   ```

10.1.6. Improving Performance for Range Searches

Range searches use operators (Section 10.4.2, "Using Operators in Search Filters") to set a bracket to search for and return an entire subset of entries within the directory. For example, this searches for every entry modified at or after midnight on January 1:

```
(modifyTimestamp>=20170101010101Z)
```

The nature of a range search is that it must evaluate every single entry within the directory to see if it is within the range given. Essentially, a range search is always an all IDs search. (Performance problems all-IDs searches are covered in detail in Section 9.1.5, "Indexing Performance" .)

For most users, the look-through limit kicks in and prevents range searches from turning into an all IDs search. This improves overall performance and speeds up range search results. However, some clients or administrative users like Directory Manager may not have a look-through limit set. In that case, a range search can take several minutes to complete or even continue indefinitely.

It is possible to set a separate range look-through limit. This allows clients and administrative users to have high look-through limits while still allowing a reasonable limit to be set on potentially performance-impaired range searches.

This is configured in the ***nsslapd-rangelookthroughlimit*** attribute. The default value is **5000**, the same as the default ***nsslapd-lookthroughlimit*** attribute value.

For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: add
add: nsslapd-rangelookthroughlimit
nsslapd-rangelookthroughlimit: 7500
```

10.2. FINDING ENTRIES USING THE DIRECTORY SERVER CONSOLE

Users can browse the **Directory** tab of the Directory Server Console to see the contents of the directory tree and search for specific entries in the directory.

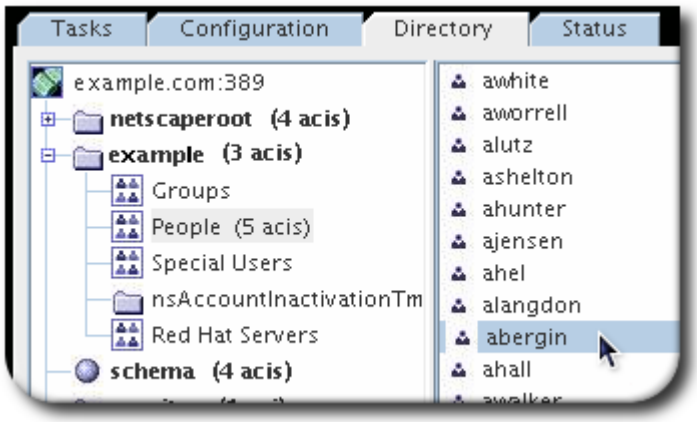


Figure 10.1. Browsing Entries in the Directory Tab

Depending on the DN used to authenticate to the directory, this tab displays the contents of the directory that the user account has access permissions to view. Browse through the contents of the tree, or right-click an entry, and select **Search** from the pop-up menu.

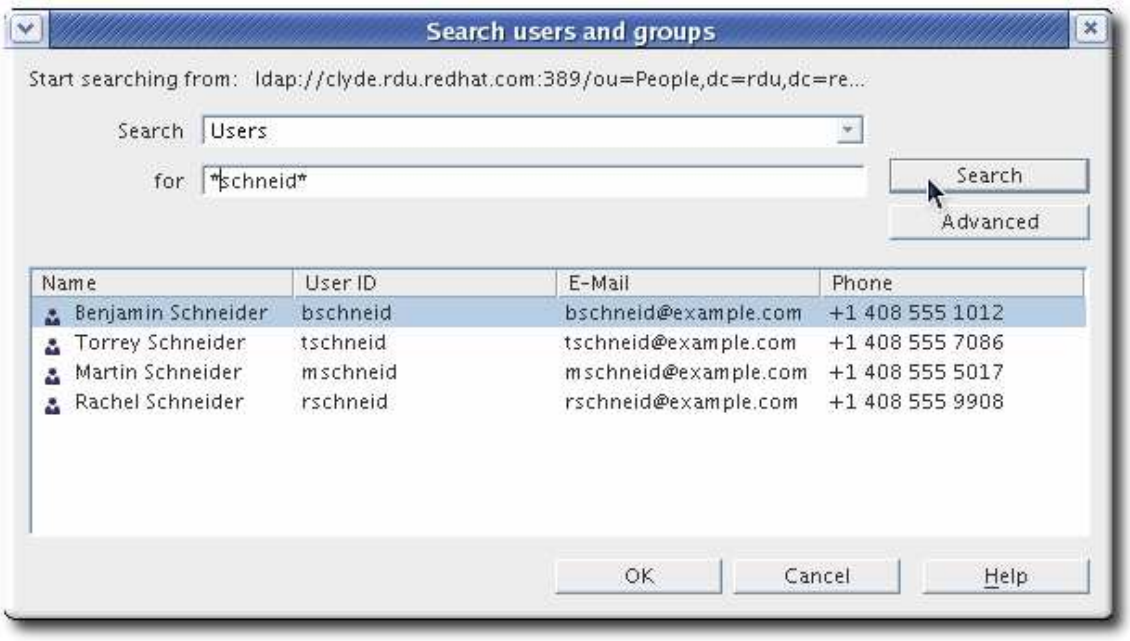


Figure 10.2. Searching for Entries

WARNING

Do not modify the contents of the **o=NetscapeRoot** suffix using the **Directory** tab unless instructed to do so by Red Hat technical support.

10.3. USING LDAPSEARCH

The **ldapsearch** command-line utility can locate and retrieve directory entries. This utility opens a connection to the specified server using the specified identity and credentials and locates entries based on a specified search filter. The search scope can include a single entry (**-s base**), an entry's immediate subentries (**-s one**), or an entire tree or subtree (**-s sub**).

NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=bjensen,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless **dc:example** has explicitly been added as an attribute in that entry.

Search results are returned in LDIF format. LDIF is defined in RFC 2849 and is described in detail in Appendix B, *LDAP Data Interchange Format*.

This section contains information about the following topics:

- Section 10.3.1, "ldapsearch Command-Line Format"
- Section 10.3.2, "Commonly Used ldapsearch Options"
- Section 10.3.3, "Using Special Characters"

10.3.1. ldapsearch Command-Line Format

The **ldapsearch** command must use the following format:

```
ldapsearch [-x | -Y mechanism] [optional_options] [optional_search_filter] [optional_list_of_attributes]
```

- Either **-x** (to disable SASL) or **-Y** (to set the SASL mechanism) must be used to configure the type of connection.
- *optional_options* is a series of command-line options. These must be specified before the search filter, if any are used.
- *optional_search_filter* is an LDAP search filter as described in Section 10.4, "LDAP Search Filters". Do not specify a separate search filter if search filters are specified in a file using the **-f** option.
- *optional_list_of_attributes* is a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see Section 10.5.6, "Displaying Subsets of Attributes" . If a list of attributes is not specified, the search returns values for all attributes permitted by the access control set in the directory (with the exception of operational attributes).

NOTE

For operational attributes to be returned as a result of a search operation, they must be explicitly specified in the search command. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (*) in the list of attributes in the **ldapsearch** command. To retrieve no attributes, just a list of the matching DNs, use the special attribute **1.1**. This is useful, for example, to get a list of DNs to pass to the **ldapdelete** command.

10.3.2. Commonly Used ldapsearch Options

The following table lists the most commonly used **ldapsearch** command-line options. If a specified value contains a space ( ), the value should be surrounded by single or double quotation marks, such as **-b "cn=My Special Group,ou=groups,dc=example,dc=com"**.

IMPORTANT

The **ldapsearch** utility from OpenLDAP uses SASL connections by default. To perform a simple bind or to use TLS/SSL, use the **-x** argument to disable SASL and allow other connection methods.

Option	Description
-b	Specifies the starting point for the search. The value specified here must be a distinguished name that currently exists in the database. This is optional if the **LDAP_BASEDN** environment variable has been set to a base DN. The value specified in this option should be provided in single or double quotation marks. For example: `-b "cn=Barbara Jensen,ou=Product Development,dc=example,dc=com"` To search the root DSE entry, specify an empty string here, such as **-b ""** .
-D	Specifies the distinguished name with which to authenticate to the server. This is optional if anonymous access is supported by the server. If specified, this value must be a DN recognized by the Directory Server, and it must also have the authority to search for the entries. For example, **-D "uid=bjensen,dc=example,dc=com"**.

Option	Description
-H	Specifies an LDAP URL to use to connect to the server. For a traditional LDAP URL, this has the following format: ldap[s]://*hostname*[:*port*] The *port* is optional; it will use the default LDAP port of 389 or LDAPS port of 636 if the port is not given. This can also use an LDAPI URL, with each element separated by the HTML hex code **%2F**, rather than a forward slash (/): ldapi://%2F*full*%2F*path*%2F*to*%2Fslapd-example.socket For LDAPI, specify the full path and filename of the LDAPI socket the server is listening to. Since this value is interpreted as an LDAP URL, the forward slash characters (/) in the path and filename must be escaped encoded as the URL escape value **%2F**. The **-H** option is used instead of **-h** and **-p**.
-h	Specifies the host name or IP address of the machine on which the Directory Server is installed. For example, **-h server.example.com**. If a host is not specified, **ldapsearch** uses the localhost.  NOTE Directory Server supports both IPv4 and IPv6 addresses.
-l	Specifies the maximum number of seconds to wait for a search request to complete. For example, **-l 300**. The default value for the ***nsslapd-timelimit*** attribute is **3600** seconds. Regardless of the value specified, **ldapsearch** will never wait longer than is allowed by the server's ***nsslapd-timelimit*** attribute.
-p	Specifies the TCP port number that the Directory Server uses. For example, **-p 1049**. The default is **389**. If **-h** is specified, **-p** must also be specified, even if it gives the default value.
-s *scope*	Specifies the scope of the search. The scope can be one of the following: **base** searches only the entry specified in the **-b** option or defined by the **LDAP_BASEDN** environment variable. **one** searches only the immediate children of the entry specified in the **-b** option. Only the children are searched; the actual entry specified in the **-b** option is not searched. **sub** searches the entry specified in the **-b** option and all of its descendants; that is, perform a subtree search starting at the point identified in the **-b** option. This is the default.

Option	Description
-w	Gives the password associated with the distinguished name that is specified in the **-D** option. If this option is not specified, anonymous access is used. For example, **-w diner892**. If there are metacharacters in the password that may be interpreted by the shell (such as exclamation points, !) then use single quotes to enclose the password. For example, **-w 'secret!'**. Alternatively, use the **-W** for the utility to prompt for the password instead of entering it in plaintext on the command line.
-x	Disables the default SASL connection to allow simple binds.
-Y *SASL_mechanism*	Defines the SASL mechanism to use for connections. For example, **-Y GSSAPI**. If **-x** is not used, then the **-Y** option must be used.
-z	Sets the maximum number of entries to return in response to a search request. For example, **-z 1000**. Normally, regardless of the value specified here, **ldapsearch** never returns more entries than the number allowed by the server's ***nsslapd-sizelimit*** attribute. However, this limitation can be overridden by binding as the root DN when using this command-line argument. When binding as the root DN, this option defaults to zero (**0**). The default value for the ***nsslapd-sizelimit*** attribute is **2000** entries.

10.3.3. Using Special Characters

When using the **ldapsearch** command-line utility, it may be necessary to specify values that contain characters that have special meaning to the command-line interpreter, such as space ( ), asterisk (*), or backslash (\). Enclose the value which has the special character in quotation marks (""). For example:

```
-D "cn=Barbara Jensen,ou=Product Development,dc=example,dc=com"
```

Depending on the command-line interpreter, use either single or double quotation marks. In general, use single quotation marks (') to enclose values. Use double quotation marks (") to allow variable interpolation if there are shell variables. Refer to the operating system documentation for more information.

10.4. LDAP SEARCH FILTERS

Search filters select the entries to be returned for a search operation. They are most commonly used with the **ldapsearch** command-line utility. When using **ldapsearch**, there can be multiple search filters in a file, with each filter on a separate line in the file, or a search filter can be specified directly on the command line.

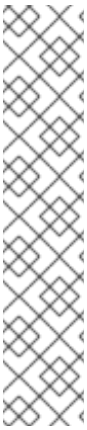
The basic syntax of a search filter is:

```
attribute operator value
```

For example:

```
buildingname>=alpha
```

In this example, ***buildingname*** is the attribute, **>=** is the operator, and **alpha** is the value. Filters can also be defined that use different attributes combined together with Boolean operators.



NOTE

When performing a substring search using a matching rule filter, use the asterisk (*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter **l** and ends with the letter **n**, enter a **l*n** in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter **u**, enter a value of **u*** in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, the asterisk must be escaped with the designated escape sequence, **\5c2a**. For example, to search for all employees with ***businessCategory*** attribute values of **Example*Net product line**, enter the following value in the search filter:

```
Example\5c2a*Net product line
```

NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=bjensen,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless **dc:example** has explicitly been added as an attribute in that entry.

10.4.1. Using Attributes in Search Filters

The most basic sort of search looks for the presence of attributes or specific values in entries. There are many variations on *how* to look for attributes in entries. It is possible to check that the attribute merely exists, to match an exact value, or to list matches against a partial value.

A *presence* search uses a wild card (an asterisk) to return every entry which has that attribute set, regardless of value. For example, this returns every entry which has a ***manager*** attribute:

```
"(manager=*)"
```

It is also possible to search for an attribute with a specific value; this is called an *equality* search. For example:

```
"(cn=babs jensen)"
```

This search filter returns all entries that contain the common name Babs Jensen. Most of the time, equality searches are not case sensitive.

When an attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match the **"(cn=babs jensen)"** filter:

```
cn: babs jensen
cn;lang-fr: babs jensen
```

It is also possible to search for a partial match on an attribute value, a *substring* index. For example:

```
"(description=*X.500*)"
"(sn=*nderson)"
"(givenname=car*)"
```

The length of the substring searches is configured in the substring index itself, as described in Section 9.6, “Changing the Width for Indexed Substring Searches”.

10.4.2. Using Operators in Search Filters

Operators in search filters set the relationship between the attribute and the given search value. For people searches, operators can be used to set a range, to return a last names within a subset of letters in the alphabet or employee numbers that come after a certain number.

```
"(employeeNumber>=500)"
"(sn~=suret)"
"(salary<=150000)"
```

Operators also enable phonetic and approximate searches, which allow more effective searches with imperfect information and are particularly useful in internationalized directories.

The operators that can be used in search filters are listed in Table 10.2, "Search Filter Operators" . In addition to these search filters, special filters can be specified to work with a preferred language collation order. For information on how to search a directory with international charactersets, see Section D.4, "Searching an Internationalized Directory".

Table 10.2. Search Filter Operators

Search Type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, **cn=Bob Johnson**
Substring	=*string** *string*	Returns entries containing attributes containing the specified substring. For example, **cn=Bob* cn=*Johnson cn=*John* cn=B*John**. The asterisk (*) indicates zero (0) or more characters.
Greater than or equal to	>=	Returns entries containing attributes that are greater than or equal to the specified value. For example, **buildingname >= alpha**.
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, **buildingname <= alpha**.
Presence	=*	Returns entries containing one or more values for the specified attribute. For example, **cn=* telephoneNumber=* manager=***.
Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example, **cn~=suret l~=san fransico** could return**cn=sarette l=san francisco**.

10.4.3. Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

```
(Boolean-operator(filter)(filter)(filter)...)
```

Boolean-operator is any one of the Boolean operators listed in Table 10.3, "Search Filter Boolean Operators" .

FOr example, this filter returns all entries that do not contain the specified value:

```
(!(cn=Ray Kultgen))
(!(objectClass=person))
```

Obviously, compound search filters are most useful when they are nested together into completed expressions:

```
(Boolean-operator(filter)((Boolean-operator(filter)(filter)))
```

These compound filters can be combined with other types of searches (approximate, substring, other operators) to get very detailed results. For example, this filter returns all entries whose organizational unit is **Marketing** and whose description field does not contain the substring **X.500**:

```
(&(ou=Marketing)(!(description=*X.500*)))
```

That filter can be expanded to return entries whose organizational unit is **Marketing**, that do not have the substring **X.500**, and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(!(description=*X.500*))(|(manager=cn=Julie Fulmer,ou=Marketing,dc=example,dc=com)
(manager=cn=Cindy Zwaska,ou=Marketing,dc=example,dc=com)))
```

This filter returns all entries that do not represent a person and whose common name is similar to **printer3b**:

```
(&(!(objectClass=person))(cn~=printer3b))
```

Table 10.3. Search Filter Boolean Operators

Operator	Symbol	Description
AND	&	All specified filters must be true for the statement to be true. For example, *(&(filter)(filter)(filter)...)*.
OR	|	At least one specified filter must be true for the statement to be true. For example, *(|(filter)(filter)(filter)...)*
NOT	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the **NOT** operator. For example, *(!(filter))*.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.

10.4.4. Using Matching Rules

A *matching rule* tells the Directory Server how to compare two values (the value stored in the attribute and the value in the search filter). A matching rule also defines how to generate index keys. Matching rules are somewhat related to attribute syntaxes. Syntaxes define *the format* of an attribute value; matching rules define how that format is compared and indexed.

There are three different types of matching rules:

- **EQUALITY** specifies how to compare two values for an equal match. For example, how to handle strings like “Fred” and “FRED”. Search filters that test for equality (e.g. *attribute=value*) use the EQUALITY rule. Equality (eq) indexes use the EQUALITY rule to generate the index keys. Update operations use the EQUALITY rule to compare values to be updated with values already in an entry.
- **ORDERING** specifies how to compare two values to see if one value is greater or less than another value. Search filters that set a range (e.g. *attribute<=value* or *attribute>=value*) use the ORDERING rule. An index for an attribute with an ORDERING rule orders the equality values.
- **SUBSTR** specifies how to do substring matching. Substring search filters (e.g. *attribute=*partial_string** or *attribute=*end_string*) use the SUBSTR rule. Substring (sub) indexes use the SUBSTR rule to generate the index keys.

IMPORTANT

A matching rule is required in order to support searching or indexing for the corresponding search filter or index type. For example, an attribute must have an EQUALITY matching rule in order to support equality search filters and eq indexes for that attribute. An attribute must have both an ORDERING matching rule and an EQUALITY matching rule in order to support range search filters and indexed range searches.

A search operation will be rejected with PROTOCOL_ERROR or UNWILLING_TO_PERFORM if an attempt is made to use a search filter for an attribute that has no corresponding matching rule.

Example 10.1. Matching Rules and Custom Attributes

Example Corp. administrators create a custom attribute type called ***MyFirstName*** with IA5 String (7-bit ASCII) syntax and an EQUALITY matching rule of caseExactIA5Match. An entry with a ***MyFirstName*** value of **Fred** is returned in a search with a filter of **(MyFirstName=Fred)**, but it is not returned for filters like **(MyFirstName=FRED)** and **(MyFirstName=fred) Fred**, **FRED**, and **fred** are all valid IA5 String values, but they do not match using the caseExactIA5Match rule.

For all three variants of Fred to be returned in a search, then the ***MyFirstName*** should be defined to use the caseIgnoreIA5Match matching rule.

An extensible matching rule search filter can be used to search for an attribute value with a different matching rule than the one defined for the attribute. The matching rule must be compatible with the syntax of the attribute being searched. For example, to run a case insensitive search for an attribute that has a case-sensitive matching rule defined for it, specify a case insensitive matching rule in the search filter.

```
(MyFirstName:caseIgnoreIA5Match:=fred)
```

NOTE

Matching rules are used for searches in internationalized directories, to specify the language types to use for the results. This is covered in Section D.4, “Searching an Internationalized Directory”.

NOTE

An index for an attributes uses whatever matching rules are defined for that attribute in its schema definition. Additional matching rules to use for an index can be configured using the ***nsMatchingRule*** attribute, as in Section 9.2.2, “Creating Indexes from the Command Line”.

The syntax of the matching rule filter inserts a matching rule name or OID into the search filter:

```
attr:matchingRule:=value
```

- *attr* is an attribute belonging to entries being searched, such as ***cn*** or ***mail***.
- *matchingRule* is a string that contains the name or OID of the rule to use to match attribute values according to the required syntax.
- *value* is either the attribute value to search for or a relational operator plus the attribute value to search for. The syntax of the value of the filter depends on the matching rule format used.

A matching rule is actually a schema element, and, as with other schema elements is uniquely identified by an object identifier (OID).

Many of the matching rules defined for Red Hat Directory Server relate to language codes and set internationalized collation orders supported by the Directory Server. For example, the OID **2.16.840.1.113730.3.3.2.17.1** identifies the Finnish collation order.

NOTE

Unlike other schema elements, additional matching rules cannot be added to the Directory Server configuration.

Most of the matching rules list in Table 10.4, “General Syntax Matching Rules” are used for equality indexes. Matching rules with *ordering* in their name are used for ordering indexes, and those with *substring* in their name are used for

substring (SUBSTR) indexes. (The matching rules used for international matching and collation orders use a different naming scheme.)

Table 10.4. General Syntax Matching Rules

Matching Rule	Object Identifiers (OIDs)	Definitions	Compatible Syntaxes
Bitwise AND Match	1.2.840.113556.1.4.803	Performs bitwise AND matches.	Typically used with:[a] Integer Numeric String
Bitwise OR Match	1.2.840.113556.1.4.804	Performs bitwise OR matches.	Typically used with:[a] Integer Numeric String
booleanMatch	2.5.13.13	Evaluates whether the values to match are TRUE or FALSE.	Boolean
caseExactIA5Match	1.3.6.1.4.1.1466.109.114.1	Makes a case-sensitive comparison of values.	IA5 Syntax URI
caseExactMatch	2.5.13.5	Makes a case-sensitive comparison of values.	Directory String Printable String OID
caseExactOrderingMatch	2.5.13.6	Allows case-sensitive ranged searches (less than and greater than).	Directory String Printable String OID
caseExactSubstringsMatch	2.5.13.7	Performs case-sensitive substring and index searches.	Directory String Printable String OID
caseIgnoreIA5Match	1.3.6.1.4.1.1466.109.114.2	Performs case-insensitive comparisons of values.	IA5 Syntax URI
caseIgnoreIA5SubstringsMatch	1.3.6.1.4.1.1466.109.114.3	Performs case-insensitive searches on substrings and indexes.	IA5 Syntax URI
caseIgnoreListMatch	2.5.13.11	Performs case-insensitive comparisons of values.	Postal Address

Matching Rule	Object Identifiers (OIDs)	Definitions	Compatible Syntaxes
caseIgnoreListSubstringsMatch	2.5.13.12	Performs case-insensitive searches on substrings and indexes.	Postal Address
caseIgnoreMatch	2.5.13.2	Performs case-insensitive comparisons of values.	Directory String Printable String OID
caseIgnoreOrderingMatch	2.5.13.3	Allows case-insensitive ranged searches (less than and greater than).	Directory String Printable String OID
caseIgnoreSubstringsMatch	2.5.13.4	Performs case-insensitive searches on substrings and indexes.	Directory String Printable String OID
distinguishedNameMatch	2.5.13.1	Compares distinguished name values.	Distinguished name (DN)
generalizedTimeMatch	2.5.13.27	Compares values that are in a Generalized Time format.	Generalized Time
generalizedTimeOrderingMatch	2.5.13.28	Allows ranged searches (less than and greater than) on values that are in a Generalized Time format.	Generalized Time
integerMatch	2.5.13.14	Evaluates integer values.	Integer
integerOrderingMatch	2.5.13.15	Allows ranged searches (less than and greater than) on integer values.	Integer
keywordMatch	2.5.13.33	Compares the given search value to a string in an attribute value.	Directory String
numericStringMatch	2.5.13.8	Compares more general numeric values.	Numeric String
numericStringOrderingMatch	2.5.13.9	Allows ranged searches (less than and greater than) on more general numeric values.	Numeric String
numericStringSubstringMatch	2.5.13.10	Compares more general numeric values.	Numeric String
objectIdentifierMatch	2.5.13.0	Compares object identifier (OID) values.	OID

Matching Rule	Object Identifiers (OIDs)	Definitions	Compatible Syntaxes
octetStringMatch	2.5.13.17	Evaluates octet string values.	Octet String
octetStringOrderingMatch	2.5.13.18	Supports ranged searches (less than and greater than) on a series of octet string values.	Octet String
telephoneNumberMatch	2.5.13.20	Evaluates telephone number values.	Telephone Number
telephoneNumberSubstringsMatch	2.5.13.21	Performs substring and index searches on telephone number values.	Telephone Number
uniqueMemberMatch	2.5.13.23	Compares both name and UID values.	Name and Optional UID
wordMatch	2.5.13.32	Compares the given search value to a string in an attribute value. This matching rule is case-insensitive.	Directory String

[a] This has a special format; the value is converted to integer before being used by Directory Server.

Table 10.5. Language Ordering Matching Rules

Matching Rule	Object Identifiers (OIDs)
English (Case Exact Ordering Match)	2.16.840.1.113730.3.3.2.11.3
Albanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.44.1
Arabic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.1.1
Belorussian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.2.1
Bulgarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.3.1
Catalan (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.4.1
Chinese - Simplified (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.49.1
Chinese - Traditional (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.50.1
Croatian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.22.1
Czechoslovakian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.5.1
Danish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.6.1
Dutch (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.33.1
Dutch - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.34.1
English - US (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.11.1
English - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.12.1

Matching Rule	Object Identifiers (OIDs)
English - Irish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.14.1
Estonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.16.1
Finnish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.17.1
French (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.18.1
French - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.19.1
French - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.20.1
French - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.21.1
German (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.7.1
German - Austrian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.8.1
German - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.9.1
Greek (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.10.1
Hebrew (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.27.1
Hungarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.23.1
Icelandic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.24.1
Italian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.25.1
Italian - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.26.1
Japanese (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.28.1
Korean (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.31.1
Lithuanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.30.1
Macedonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.32.1
Norwegian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.35.1
Norwegian - Bokmul (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.36.1
Norwegian - Nynorsk (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.37.1
Polish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.38.1
Romanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.39.1
Russian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.40.1
Serbian - Cyrillic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.45.1
Serbian - Latin (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.41.1

Matching Rule	Object Identifiers (OIDs)
Slovakian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.42.1
Slovenian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.43.1
Spanish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.15.1
Swedish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.46.1
Turkish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.47.1
Ukrainian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.48.1

Table 10.6. Language Substring Matching Rules

Matching Rule	Object Identifiers (OIDs)
English (Case Exact Substring Match)	2.16.840.1.113730.3.3.2.11.3.6
Albanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.44.1.6
Arabic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.1.1.6
Belorussian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.2.1.6
Bulgarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.3.1.6
Catalan (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.4.1.6
Chinese - Simplified (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.49.1.6
Chinese - Traditional (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.50.1.6
Croatian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.22.1.6
Czechoslovakian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.5.1.6
Danish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.6.1.6
Dutch (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.33.1.6
Dutch - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.34.1.6
English - US (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.11.1.6
English - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.12.1.6
English - Irish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.14.1.6
Estonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.16.1.6
Finnish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.17.1.6
French (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.18.1.6
French - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.19.1.6
French - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.20.1.6

Matching Rule	Object Identifiers (OIDs)
French - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.21.1.6
German (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.7.1.6
German - Austrian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.8.1.6
German - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.9.1.6
Greek (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.10.1.6
Hebrew (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.27.1.6
Hungarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.23.1.6
Icelandic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.24.1.6
Italian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.25.1.6
Italian - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.26.1.6
Japanese (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.28.1.6
Korean (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.29.1.6
Latvian, Lettish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.31.1.6
Lithuanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.30.1.6
Macedonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.32.1.6
Norwegian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.35.1.6
Norwegian - Bokmul (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.36.1.6
Norwegian - Nynorsk (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.37.1.6
Polish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.38.1.6
Romanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.39.1.6
Russian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.40.1.6
Serbian - Cyrillic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.45.1.6
Serbian - Latin (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.41.1.6
Slovakian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.42.1.6
Slovenian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.43.1.6
Spanish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.15.1.6
Swedish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.46.1.6
Turkish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.47.1.6
Ukrainian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.48.1.6

10.5. EXAMPLES OF COMMON LDAPSEARCHES

The next set of examples assumes the following:

- The search is for all entries in the directory.
- The directory is configured to support anonymous access for search and read. This means that no bind information has to be supplied in order to perform the search. For more information on anonymous access, see Section 13.4.2, “Defining User Access - userdn Keyword” .
- The server is located on a host named **server.example.com**.
- The server uses port number **389**. Since this is the default port, the port number does not have to be sent in the search request.
- SSL is enabled for the server on port **636** (the default SSL port number).
- The suffix under which all data are stored is **dc=example,dc=com**.

10.5.1. Returning All Entries

Given the previous information, the following call will return all entries in the directory (subject to the configured size and time resource limits):

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x "
(objectclass=*)"
```

"objectclass=*" is a search filter that matches any entry in the directory. Since every entry must have an object class, and the ***objectclass*** attribute is always indexed, this is a useful search filter to return every entry.

10.5.2. Specifying Search Filters on the Command Line

A search filter can be specified directly on the command line as long as the filter is enclosed in quotation marks ("filter"). If the filter is supplied with the command, do not specify the **-f** option. For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
"cn=babs jensen"
```

10.5.3. Searching the Root DSE Entry

The root DSE is a special entry that contains information about the directory server instance, including all of the suffixes supported by the local Directory Server. This entry can be searched by supplying a search base of "", a search scope of **base**, and a filter of **"objectclass=*"**. For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b "" -s base "objectclass=*"
```

10.5.4. Searching the Schema Entry

The **cn=schema** entry is a special entry that contains information about the directory schema, such as object classes and attribute types.

The following command lists the content of the **cn=schema** entry:

```
# ldapsearch -o ldif-wrap=no -D "cn=directory manager" -W -b "cn=schema" \
   '(objectClass=subSchema)' -s sub objectClasses attributeTypes matchingRules \
   matchingRuleUse  dITStructureRules nameForms ITContentRules ldapSyntaxes
```

10.5.5. Using LDAP_BASEDN

To make searching easier, it is possible to set the search base using the **LDAP_BASEDN** environment variable. Doing this means that the search base does not have to be set with the **-b** option. For information on how to set environment variables, see the documentation for the operating system.

Typically, set **LDAP_BASEDN** to the directory's suffix value. Since the directory suffix is equal to the root, or topmost, entry in the directory, this causes all searches to begin from the directory's root entry.

For example, set **LDAP_BASEDN** to **dc=example,dc=com** and search for **cn=babs jensen** in the directory, use the following command-line call:

```
export LDAP_BASEDN="dc=example,dc=com"

ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x "cn=babs jensen"
```

In this example, the default scope of **sub** is used because the **-s** option was not used to specify the scope.

10.5.6. Displaying Subsets of Attributes

The **ldapsearch** command returns all search results in LDIF format. By default, **ldapsearch** returns the entry's distinguished name and all of the attributes that a user is allowed to read. The directory access control can be set such that users are allowed to read only a subset of the attributes on any given directory entry. Only operational attributes are not returned. For operational attributes to be returned as a result of a search operation, explicitly specify them in the search command.

It may not be necessary to have all of the attributes for an entry returned in the search results. The returned attributes can be limited to just a few specific attributes by specifying the desired ones on the command line immediately after the search filter. For example, to show the ***cn*** and ***sn*** attributes for every entry in the directory, use the following command-line call:

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x "
(objectclass=*)" sn cn
```

10.5.7. Searching for Operational Attributes

Operational attributes are special attributes set by the Directory Server itself that are used by the server to perform maintenance tasks, like processing access control instructions. They also show specific information about the entry, like the time it was initially created and the name of the user who created it. Operational attributes are available for use on every entry in the directory, regardless of whether the attribute is specifically defined for the object class of the entry.

Operational attributes are not returned in regular **ldapsearch**es, so to return operational attributes, they have to be explicitly specified in the **ldapsearch** request.

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x "
(objectclass=*)" creatorsName createTimestamp modifiersName modifyTimestamp
```

The complete list of operational attributes is in the "Operational Attributes and Object Classes" chapter in the *Red Hat Directory Server 9 Configuration, Command, and File Reference* .

NOTE

To return all of the regular entry attributes along with the specified operational attributes, use the special search attribute, **"*"**, in addition to the operational attributes that are listed.

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -
s sub -x "(objectclass=*)" "*" aci
```

The asterisk must be enclosed in quotation marks to prevent it from being interpreted by the shell.

10.5.8. Specifying Search Filters Using a File

Search filters can be entered into a file instead of entering them on the command line. In this case, specify each search filter on a separate line in the file. The **ldapsearch** command runs each search in the order in which it appears in the file.

For example:

```
sn=Francis
givenname=Richard
```

ldapsearch first finds all the entries with the surname **Francis**, then all the entries with the givenname **Richard**. If an entry is found that matches both search criteria, then the entry is returned twice.

For example, in this search, the filters are specified in a file named **searchdb**:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -f searchdb
```

The set of attributes returned here can be limited by specifying the attribute names at the end of the search line. For example, the following **ldapsearch** command performs both searches but returns only the DN and the ***givenname*** and ***sn*** attributes of each entry:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -f searchdb sn givenname
```

10.5.9. Specifying DNs That Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, the comma must be escaped with a backslash (\). For example, to find everyone in the **example.com Bolivia, S.A.** subtree, use the following command:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s base -b
"l=Bolivia\,S.A.,dc=example,dc=com" "objectclass=*"
```

10.5.10. Using Client Authentication When Searching

Client authentication uses a stored certificate to bind to the directory rather than simple user name and password combination. The SSL parameters are set separately as an environment variable or by editing **ldap.conf**. Then, the **ldapsearch** can be run by disabling SASL and specifying the SSL port. For example:

```
export LDAPTLS_CACERTDIR=/etc/dirsrv/slapd-instance_name
export LDAPTLS_CERT=Server-Cert
export LDAPTLS_KEY=internal:secret

ldapsearch -h server.example.com -p 636 -b "dc=example,dc=com" -x "givenname=Richard"
```

The possible environment variables are described more in Section A.1, “Environment Variables Used with LDAP Client Tools”.

10.5.11. Searching with Language Matching Rules

To explicitly submit a matching rule in a search filter, insert the matching rule after the attribute:

```
attr:matchingRule:=value
```

Matching rules are frequently used for searching internationalized directories. For example, this seearches for the department numbers after N4709 in the Swedish (**2.16.840.1.113730.3.3.2.46.1**) matching rule.

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

More examples of performing internationalized searches are given in Section D.4, “Searching an Internationalized Directory”.

10.5.12. Searching for Attributes with Bit Field Values

Bitwise searches use the bitwise AND or bitwise OR matching rules to perform bitwise search operaions on attributes with values that are bit fields.

NOTE

Attributes with values for bit fields are not common in LDAP. (No default Directory Server schema use bit fields as attribute syntax.) However, several LDAP syntaxes support interger-style values. Custom attributes can be defined which use bit field values, and applications can use those custom attributes to perform bitwise operations against bit field values.

The bitwise AND matching rule (**1.2.840.113556.1.4.803**) checks that the bit given in the assertion value is set in the bit field attribute value. (This is somewhat analogous to an equality search.) In this example, the userAccountControl value must be set to the bit representing 2.

```
"(UserAccountControl:1.2.840.113556.1.4.803:=2)"
```

In this example, the userAccountControl value must have all of the bits set that are set in the value 6 (bits 2 and 4).

```
"(UserAccountControl:1.2.840.113556.1.4.803:=6)”
```

The bitwise OR matching rule (**1.2.840.113556.1.4.804**) checks to see if *any* of the bits in the assertion string are represented in the attribute value. (This is somewhat analogous to a substring search.) In this example, the userAccountControl value must have any of the bits which are set in the bit field of 6, meaning that the attribute value can be 2, 4, or 6.

```
"(UserAccountControl:1.2.840.113556.1.4.804:=6)"
```

Bitwise searches can be used with Windows-Red Hat Enterprise Linux integration, such as using Samba file servers.

NOTE

Microsoft has good documentation on bitwise operators at http://msdn.microsoft.com/en-us/library/aa746475.

10.6. USING PERSISTENT SEARCH

A persistent search is an **ldapsearch** which remains open even after the initial search results are returned.

IMPORTANT

The OpenLDAP client tools with Red Hat Enterprise Linux do not support persistent searches. The server itself, however, does. Other LDAP clients must be used to perform persistent searches.

The purpose of a persistent search is to provide a continuous list of changes to the directory entries as well as the complete entries themselves, something like a hybrid search and changelog. Therefore, the search command must specify what entries to return (the search parameters) and what changes cause an entry to be returned (entry change parameters).

Persistent searches are especially useful for applications or clients which access the Directory Server and provide two important benefits:

- Keep a consistent and current local cache.

  Any client will query local cache before trying to connect to and query the directory. Persistent searches provide the local cache necessary to improve performance for these clients.

- Automatically initiate directory actions.

  The persistent cache can be automatically updated as entries are modified, and the persistent search results can display what kind of modification was performed on the entry. Another application can use that output to update entries automatically, such as automatically creating an email account on a mail server for new users or generating a unique user ID number.

There are some performance considerations when running persistent searches, as well:

- The **ldapsearch** does not send a notification when the client disconnects, and the change notifications are not sent for any changes made while the search is disconnected. This means that the client's cache will not be updated if it is ever disconnected and there is no good way to update the cache with any new, modified, or deleted entries that were changed while it was disconnected.

- An attacker could open a large number of persistent searches to launch a denial of service attack.

- A persistent search requires leaving open a TCP connection between the Directory Server and client. This should only be done if the server is configured to allow a lot of client connections and has a way to close idle connections.

- Each persistent search runs on a separate thread in Directory Server.

In the access logs, a persistent search is identifies with the tag **options=persistent**.

```
[12/Jan/2009:12:51:54 -0500] conn=19636710736396323 op=0 SRCH base="dc=example,dc=com" scope=2
filter="(objectClass=person)" attrs=ALL options=persistent
```

10.7. SEARCHING WITH SPECIFIED CONTROLS

The Directory Server has defined controls in its ***supportedControls*** attribute in its DSE. Some of these define server operations like replication; other are allowed extended operations like get effective rights or dereferencing controls which clients can pass through LDAP operations to the server.

These controls can be specified using the **-E** option by giving the control OID, its criticality for the **ldapsearch**, and any information required for the control operation.

```
-E '[!]control_OID:control_information'
```

Some controls, like server-side sorting and simple paged results, have an alias that can be used to pass the control to the search operation. When the control alias is used, then the results are formatted, since the control is recognized by the client.

10.7.1. Retrieving Effective User Rights

A get effective-rights search control is passed using the control OID. For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
-E '!1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=jsmith,ou=people,dc=example,dc=com' "(objectclass=*)"
```

IMPORTANT

When a control is passed with its OID, the results from the search are unformatted.

Get effective rights searches are covered in much more detail in the access control chapter, Section 13.7, “Checking Access Rights on Entries (Get Effective Rights)”.

10.7.2. Using Server-Side Sorting

Server-side sorting is performed as other control operations, using the **-E** flag and the **sss** control alias. The structure of the operation sets the attribute by which to sort the results and, optionally, the sort order and ordering rule.

```
-E sss=[-]attribute_name:[ordering_rule_OID]
```

The dash (**-**) is an optional flag that reverses the sort order, which naturally runs descending. The matching rule tables in Section 10.4.4, “Using Matching Rules” contain the ordering rules supported by the Directory Server.

For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
-E sss=-uidNumber:2.5.13.15 "(objectclass=*)"
```

10.7.3. Performing Dereferencing Searches

A *dereferencing* search is a quick way to track back over cross-references in an entry and return information about the referenced entry. For example, a group entry contains references to its member's user entries. A regular search first searches for the group, then lists its members, and then requires a separate search for each member. A dereferencing search for the group entry returns information about the members – such as their locations, email addresses, or managers – *along with* the information for the group, all in a single search request.

Dereferencing simplifies many client operations and reduces the number of search operations that are performed. Cross-links show relationships between entries. Some operations may require getting a list of cross-links from one entry and then performing a series of subsequent searches to get information from each entry on the list. Dereferencing allows those sequences of searches to be consolidated into a single search.

IMPORTANT

Dereferencing operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support dereferencing searches.

The format of the dereference arguments is:

```
-E 'deref=deref_attribute:list_of_attributes'
```

The *deref_attribute* is the attribute in the search target that contains the reference. This can be any attribute which has a DN for its value, such as **member** or **manager**.

NOTE

Not only must the value of the *deref_attribute* be a DN, but the actual defined syntax for the attribute must be DN syntax (**1.3.6.1.4.1.1466.115.121.1.12**).

The *list_of_attributes* is one or more attributes in the referenced entry which will be returned along with the primary search results. Multiple attributes can be separated by commas, like **l,mail,cn**.

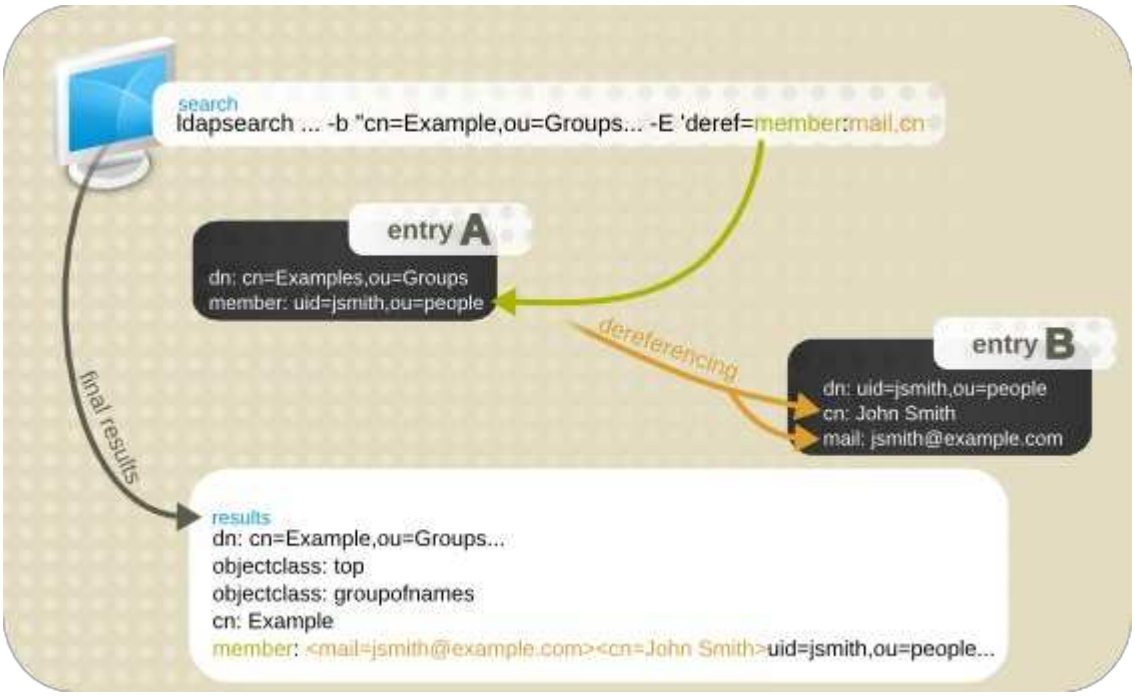


Figure 10.3. Simple Dereferencing Search Command

The requested dereferenced information requested in the search argument is returned with the rest of the search results. For example, this dereferencing search tells the server to use the ***member*** attribute in the search target entry (the Engineers group) as the *deref_attribute*. It then returns the locality attribute for each member.

```
ldapsearch -x -D "cn=Directory Manager" -W -b "cn=Example,ou=Groups,dc=example,dc=com" -E
'deref=member:mail,cn' "(objectclass=*)"

# Engineers, Groups, example.com
dn: cn=Engineers,ou=Groups,dc=example,dc=com
control: 1.3.6.1.4.1.4203.666.5.16 false MIQAAADNMIQAAAA1BAZtZW1iZXIEK2NuPURld
 mVsb3BlcnMsIG91PUdyb3VwcywgZGM9ZXhhbXBsZSxkYz1jb20whAAAADIEBm1lbWJlcgQoY249VG
 VzdGVycywgb3U9R3JvdXBzLCBkYz1leGFtcGxlLGRjPWNvbTCEAAAAVAQGbWVtYmVyBCp1aWQ9ZW5
 nLCBvdT1lbmdpbmVlcmluZywgZGM9ZXhhbXBsZSxkYz1jb22ghAAAABowhAAAABQEAWwxhAAAAsE
 CUNhbWJyaWRnZQ==
# member: <mail=jsmith@example.com><cn=John Smith>;uid=jsmith,ou=people,dc=example,dc=com
 objectClass: top
objectClass: inetuser
objectClass: groupofnames
cn: Engineers
member: uid=jsmith,ou=people,dc=example,dc=com
```

10.7.4. Using Simple Paged Results

Search results can be very large, and a part of processing the results is organizing the results. One method of doing this is using *simple paged results*, a control that breaks the results into pages of a certain length.

The simple paged results control sets the number of entries to display at a time. The results can be scrolled through one page at a time which makes the results easier to digest. The full behavior of the control is described in RFC 2696.

Simple paged results are implemented as an LDAP control extension for the Directory Server. Its OID is **1.2.840.113556.1.4.319**.

How Simple Paged Results Work

When you start a simple paged results search:

1. The client sends the search to the server, together with the paged results control and with how many records to return in the first page.
2. Before Directory Server starts returning data, the server generates an estimate how many records can be returned in total.

   The estimate of records is not an exact number. The total number of records returned can be lower than the estimate. The reasons for such a scenario include

   - attributes used in the search filter do not exist in the index. For an optimal result, all queried attributes must be indexed.
   - before an entry is send to the client, access control lists (ACL) are validated. Insufficient permissions can prevent the entry from being returned.

   After generating the estimate, the server sends the first set of results, a cookie, and the estimated number of records.
3. The returned records are displayed in the client. The user can now enter how many records should be returned in the next request. The requested number is now sent, together with the cookie, to the server.
4. The server retrieves the requested number of records from the database and sends them together with a cookie to the client.
5. The previous two steps are repeated until all records are sent or the search is cancelled.

Simple Paged Results and OpenLDAP Tools

The format of the simple paged result search option with **ldapsearch** is:

```
-E pg=size
```

The *size* value is the page size, or the number of entries to include per page. For example:

```
ldapsearch -x -D "cn=Directory Manager" -W -b "ou=Engineers,ou=People,dc=example,dc=com" -E pg=3 "
(objectclass=*)" cn

dn: uid=jsmith,ou=Engineers,ou=People,dc=example,dc=com
  cn: John Smith

dn: uid=bjensen,ou=Engineers,ou=People,dc=example,dc=com
  cn: Barbara Jensen

dn: uid=hmartin,ou=Engineers,ou=People,dc=example,dc=com
  cn: Henry Martin

Results are sorted.
next page size (3): 5
```

The tag at the end shows the configured page size (the number in parentheses) from the search. After the colon, one enters the page size for the next page, so entering **5** as shown would open the next page of results with five entries.

IMPORTANT

Simple paged results operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support simple paged results, such as Perl Net::LDAP.

Simple Paged Results and Server-Side Sorting

Simple paged results can be used together with server-side sorting. Server-side sorting is a control which performs the sort process on the server rather than in a client; this is usually done for a search which uses a particular matching rule. (This behavior is defined in RFC 2891.) The OpenLDAP client tools do not support server-side sort with the simple paged results control, but other LDAP utilities such as Perl Net::LDAP do support both.

Multiple Simple Paged Results Requests on a Single Connection

Some clients may open a single connection to the Directory Server, but send multiple operation requests, including multiple search requests using the simple paged results extension.

Directory Server can manage and interpret multiple simple paged searches. Each search is added as an entry in an array. When the paged search request is first sent, there is a cookie created and associated with the search results. Each page of results is returned with that cookie, and that cookie is used to request the next page of results. On the

last page, the cookie is empty, signalling the end of the results. This keeps each set of search results separate.

When there are multiple simple paged results on a single connection, the timeout limits are still observed, but *all* open search requests much reach their configured time limit before *any* paged search is disconnected.

Simple Paged Results, Contrasted with VLV Indexes

VLV indexes are similar to simple paged results in that they also return a usable browsing list of results. The main difference is in how that list is generated. Simple paged results are calculated per search, while VLV indexes are a permanent list. Overall, VLV indexes are faster for searches, but do require some server-side configuration and overhead for the server to maintain.

NOTE

Simple paged results and VLV indexes *cannot* be used on the same search. Simple paged results would attempt to manipulate the VLV index, which is already a browsing index. If the control is passed for a search using a VLV index, then the server returns an **UNWILLING_TO_PERFORM** error.

For more information on VLV indexes, see Section 9.4, “Creating Browsing (VLV) Indexes” .

10.7.5. Pre- and Post-read Entry Response Controls

Red Hat Directory Server supports pre- and post-read entry response controls according to RFC 4527. If a client requests one or both response controls, an LDAP search entry is returned, that contains the attribute's value before and after the update.

When the pre-read control is used, an LDAP search query is returned containing the specified attribute's value before modification. When the post-read control is used, the query contains the attribute's value after modification. Both controls can be used at the same time. For example, to update the ***description*** attribute and display the value before and after the modification:

```
# ldapmodify -D "cn=directory manager" -W -x \
    -e \!preread=description -e \!postread=description
dn: uid=user,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: new description

modifying entry "uid=user,ou=People,dc=example,dc=com"
control: 1.3.6.1.1.13.1 false ZCkEJXVpZD1qdXNlcixvdT1QZW9wbGUsZGM9ZXhhbXBsZSxk
 Yz1jb20wAA==
# ==> preread
dn: uid=user,ou=People,dc=example,dc=com
description: old description
# <== preread
control: 1.3.6.1.1.13.2 false ZEsEJXVpZD1qdXNlcixvdT1QZW9wbGUsZGM9ZXhhbXBsZSxk
Yz1jb20wIjAgBAtkZXNjcmlwdGlvbjERBA9uZXcgZGVzY3JpcHRpb24=
# ==> postread
dn: uid=user,ou=People,dc=example,dc=com
description: new description
# <== postread
```

CHAPTER 11. MANAGING REPLICATION

Replication is the mechanism by which directory data is automatically copied from one Red Hat Directory Server instance to another; it is an important mechanism for extending the directory service beyond a single server configuration. This chapter describes the tasks to be performed on the master and consumer servers to set up single-master replication, multi-master replication, and cascading replication.

11.1. REPLICATION OVERVIEW

Replication is the mechanism by which directory data is automatically copied from one Directory Server to another. Updates of any kind — entry additions, modifications, or even deletions — are automatically mirrored to other Directory Servers using replication.

11.1.1. What Directory Units Are Replicated

The smallest unit of of the directory which can be replicated is a database. This means that one can replicate an entire database but not a subtree within a database. Therefore, when creating the directory tree, consider any replication plans as part of determining how to distribute information.

Replication also requires that one database correspond to one suffix. This means that a suffix (or namespace) that is distributed over two or more databases using custom distribution logic cannot be replicated. For more information on this topic, see Section 2.2, "Creating and Maintaining Databases".

11.1.2. Read-Write and Read-Only Replicas

A database that participates in replication is called a *replica*. There are two kinds of replicas: read-write or read-only. A *read-write replica* contains master copies of directory information and can be updated. A *read-only replica* services read, search, and compare requests, but refers all update operations to read-write replicas. A server can hold any number of read-only or read-write replicas.

11.1.3. Suppliers and Consumers

A server that holds a replica that is copied to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is copied from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica, with two exceptions:

- In the case of cascading replication, the hub server holds a read-only replica that it supplies to consumers. Section 11.2.3, "Cascading Replication" has more information.
- In the case of multi-master replication, the *masters* are both suppliers and consumers for the same information. For more information, see Section 11.2.2, "Multi-Master Replication".

Replication is always initiated by the supplier server, never by the consumer (*supplier-initiated replication*). Supplier-initiated replication allows a supplier server to be configured to push data to multiple consumer servers.

11.1.4. Changelog

Every supplier server maintains a *changelog*, a record of all changes that a supplier or hub needs to send to its consumers. A changelog is a special kind of database that describes the modifications that have occurred on a replica. The supplier server then replays these modifications to the replicas stored on consumer servers or to other suppliers, in the case of multi-master replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the changelog.

The changelog uses the same database environment as the main database. Implementing the changelog as part of the main database ensures the database and changelog are always synchronized, reduces the required database cache size (10MB by default), and simplifies backup and restore operations.

IMPORTANT

When the database of a master server is backed up, then it should be backed up using the **db2bak.pl** Perl script or using the Directory Server Console if the server is kept running. The changelog only writes its RUV entries to the database when the server is shut down; while the server is running, the changelog keeps its changes in memory. For the Perl script and the Console, these changelog RUVs are written to the database before the backup process runs. However, that step is not performed by the command-line script.

The **db2bak** should not be run on a running master server. Either use the Perl script or stop the server before performing the backup.

In Directory Server, the changelog is only intended for internal use by the server. For other applications to read the changelog, use the Retro Changelog Plug-in, as described in Section 11.21, “Using the Retro Changelog Plug-in” .

11.1.5. Replication Identity

When replication occurs between two servers, the replication process uses a special entry, called the *replication manager* entry, to identify replication protocol exchanges and to control access to the directory data. The replication manager entry, or any entry used during replication, must meet the following criteria:

- It is created on the consumer server (or hub) and *not* on the supplier server.
- Create this entry on *every* server that receives updates from another server, meaning on every hub or dedicated consumer.
- When a replica is configured as a consumer or hub (a replica which receives updates from another server), this entry must be specified as the one authorized to perform replication updates.
- The replication agreement is created on the supplier server, the DN of this entry must be specified in the replication agreement.
- The supplier bind DN entry must not be part of the replicated database for security reasons.
- This entry, with its special user profile, bypasses all access control rules defined on the consumer server for the database involved in that replication agreement.

NOTE

In the Directory Server Console, this replication manager entry is referred to as the *supplier bind DN*, which may be misleading because the entry does not actually exist on the supplier server. It is called the supplier bind DN because it is the entry which the supplier uses to bind to the consumer. This entry actually exists, then, on the consumer.

For more information on creating the replication manager entry, see Section 11.3, “Creating the Supplier Bind DN Entry”.

11.1.6. Replication Agreement

Directory Servers use replication agreements to define their replication configuration. A replication agreement describes replication between *one* supplier and *one* consumer only. The agreement is configured on the supplier server and must specify all required replication information:

- The database to be replicated.
- The consumer server to which the data is pushed.
- The days and times during which replication can occur.
- The DN and credentials that the supplier server must use to bind (the replication manager entry or supplier bind DN).
- How the connection is secured (SSL, client authentication).
- Any attributes that will not be replicated (fractional replication).

11.1.7. Replicating a Subset of Attributes with Fractional Replication

Fractional replication sets a specific subset of attributes that will not be transmitted from a supplier to the consumer (or another supplier). Administrators can therefore replicate a database without replicating all the information that it contains or all of the information in every entry.

Fractional replication is enabled and configured per replication agreement, not per entry. Excluding attributes from replication is applied equally to all entries within the replication agreement's scope.

As far as the consumer server is concerned, the excluded attributes always have no value. Therefore, a client performing a search against the consumer server will never see the excluded attributes. Similarly, should it perform a search that specifies those attributes in its filter, no entries will match.

It is possible to set different attributes to be replicated for an incremental update and a total update. The incremental update list (***nsDS5ReplicatedAttributeList***) must always be set to enable fractional replication; if that is the only attribute set, then it applies to both incremental and total updates. The optional ***nsDS5ReplicatedAttributeListTotal*** attribute sets an additional fractional replication list for total updates. This is described in Section 11.9.1, “Setting Different Fractional Replication Attributes for Total and Incremental Updates” .

NOTE

An update to an excluded attribute still triggers a modify event and generates an empty replication update. The ***nsds5ReplicaStripAttrs*** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attribtes like ***modifiersName***.

If a replication event is *not* empty, the stripped attributes *are* replicated. These attributes are removed from updates only if the event would otherwise be emtpy.

11.1.7.1. The Replication Keep-alive Entry

When you update an attribute on a master, the change sequence number (CSN) is increased on the master. In a replication topology, this server now connects to the first consumer and compares the local CSN with the CSN on the consumer. If it is lower, the update is retrieved from the local changelog and replicated to the consumer. In a replication topology with fractional replication enabled, this can cause problems: For example, if only attributes are updated on the master that are excluded from replication, no update to replicate is found, and therefore the CSN is not updated on the consumer. In certain scenarios, such as when only attributes are updated on a master that are excluded from replication, unnecessary searching for updates on the supplier can cause other servers to receive the data later than needed . To work around this problem, Directory Server uses keep-alive entries.

If all updated attributes on the master are excluded from replication and the number of skipped updates exceeds 100, the ***keepalivetimestamp*** attribute is updated on the supplier and replicated to the consumer. Because the ***keepalivetimestamp*** attribute is not excluded from replication, the update of the keep-alive entry is replicated, the CSN on the consumer is updated, and then equal to the one on the supplier. The next time the supplier connects to the consumer, only updates that are newer than the CSN on the consumer are searched. This reduces the amount of time spent by a supplier to search for new updates to send.

The replication keep-alive entry is created on demand on a master and contains the replica ID of the master in the distinguished name (DN). Each keep-alive entry is specific to a given master. For example:

```
dn: cn=repl keep alive 14,dc=example,dc=com
objectclass: top
objectclass: ldapsubentry
objectclass: extensibleObject
cn: repl keep alive 14
keepalivetimestamp: 20170227190346Z
```

The keep-alive entry is updated in the following situations (if it does not exist before the update, it is created first):

- When a fractional replication agreement skips more than 100 updates and does not send any updates before ending the replication session.
- When a master initializes a consumer, initially it creates its own keep-alive entry. A consumer that is also a master does not create its own keep-alive entry unless it also initializes another consumer.

11.1.8. Replication with 4.x Versions of Directory Server

The replication mechanism in Directory Server 9.0 is different from the mechanism used in 4.x of Directory Server. Compatibility with Netscape 4.x and iPlanet versions of Directory Server is provided through two Directory Server plug-ins:

- *Legacy Replication Plug-in.* The Legacy Replication Plug-in makes a Directory Server 9.0 instance behave as a 4.x Directory Server in a consumer role. For information on how to implement legacy replication using this plug-in, see Section 11.20, "Replication with Earlier Releases".
- *Retro Changelog Plug-in.* The Retro Changelog Plug-in can be used for a Directory Server supplier to maintain a 4.x-style changelog. This is sometimes necessary for legacy applications that have a dependency on the Directory Server 4.x changelog format because they read information from the changelog. For more information on the Retro Changelog Plug-in, see Section 11.21, "Using the Retro Changelog Plug-in" .

NOTE

Replication with 8.x versions of Red Hat Directory Server, including multi-master replication and using 8.x replicas with 9.0 masters, is fully supported. The replication mechanisms used for Directory Server 7.x and 8.x is similar to the mechanism used in Directory Server 9.0. No additional plug-ins or configuration are required for replication to work with versions 6.x, 7.x, 8.x, or 9.x of Directory Server.

Any incompatibilities are related to enhanced features, additional schema, and other features that are available in Directory Server 9.0 which may not be available in 7.x or 8.x servers.

11.2. REPLICATION SCENARIOS

- Section 11.2.1, "Single-Master Replication"
- Section 11.2.2, "Multi-Master Replication"
- Section 11.2.3, "Cascading Replication"

These basic strategies can be combined in a variety of ways to create the best replication environment.

NOTE

Whatever replication scenario is implemented, consider schema replication. To avoid conflict resolution loops, the Referential Integrity Plug-in should only be enabled on one supplier replica in a multi-master replication environment. The plug-in is off by default.

11.2.1. Single-Master Replication

In the simplest replication scenario, the master copy of directory data is held in a single read-write replica on one server called the *supplier server*. The supplier server also maintains changelog for this replica. On another server, called the *consumer server*, there can be multiple read-only replicas. Such scenarios are called *single-master configurations*. Figure 11.1, "Single-Master Replication" shows an example of single-master replication.

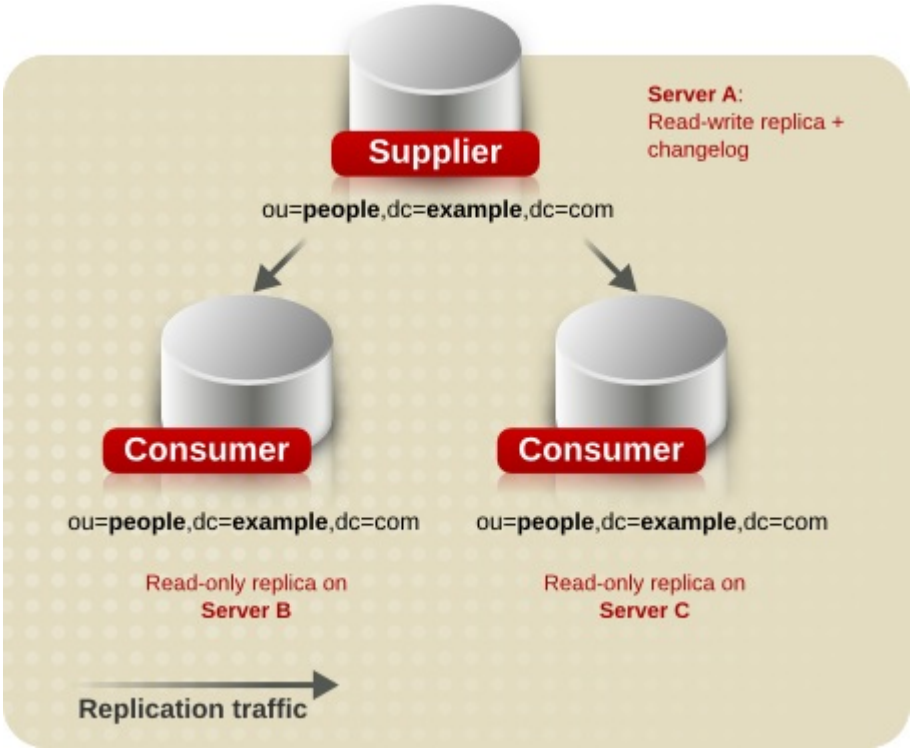


Figure 11.1. Single-Master Replication

In this particular configuration, the **ou=people,dc=example,dc=com** suffix receives a large number of search requests. Therefore, to distribute the load, this tree, which is mastered on Server A, is replicated to two read-only replicas located on Server B and Server C.

For information on setting up a single-master replication environment, see Section 11.4, "Configuring Single-Master Replication".

11.2.2. Multi-Master Replication

Directory Server also supports complex replication scenarios in which the same suffix (database) can be mastered on many servers. This suffix is held in a read-write replica on each server. This means that each server maintains a changelog for the read-write replica.

Multi-master replication in Directory Server supports as many as 20 masters, an unlimited number of hub suppliers, and an unlimited number of consumer servers. Each consumer server holds a read-only replica. The consumers can receive updates from any or all the suppliers. The consumers also have referrals defined for all the suppliers to forward any update requests that the consumers receive. Such scenarios are called *multi-master configurations*.

Figure 11.2, "Multi-Master Replication (Two Masters)" shows an example of multi-master replication scenario with two supplier servers and two consumer servers.

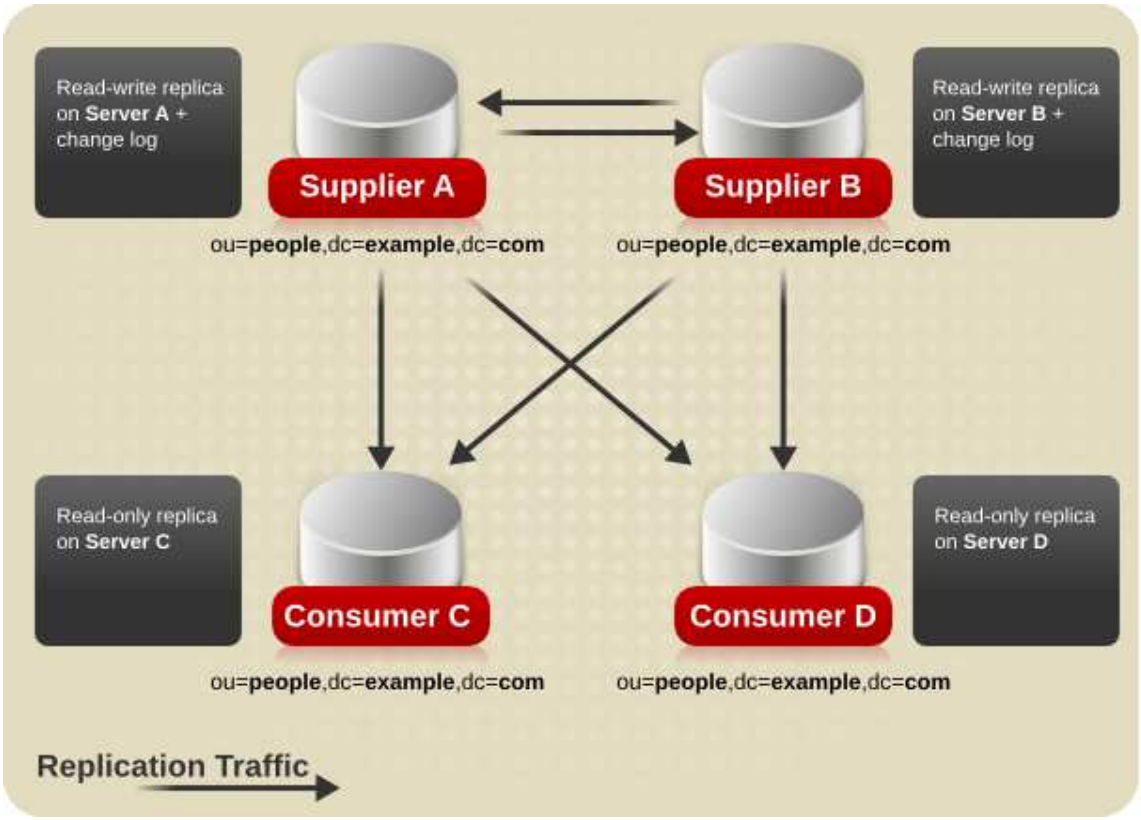


Figure 11.2. Multi-Master Replication (Two Masters)

Figure 11.3, "Multi-Master Replication (Four Masters)" shows a sample of multi-master replication scenario with four supplier servers and eight consumer servers. In this sample setup, each supplier server is configured with ten replication agreements to feed data to two other supplier servers and all eight consumer servers. (The Directory Server can have as many as 20 masters in a multi-master setup.)

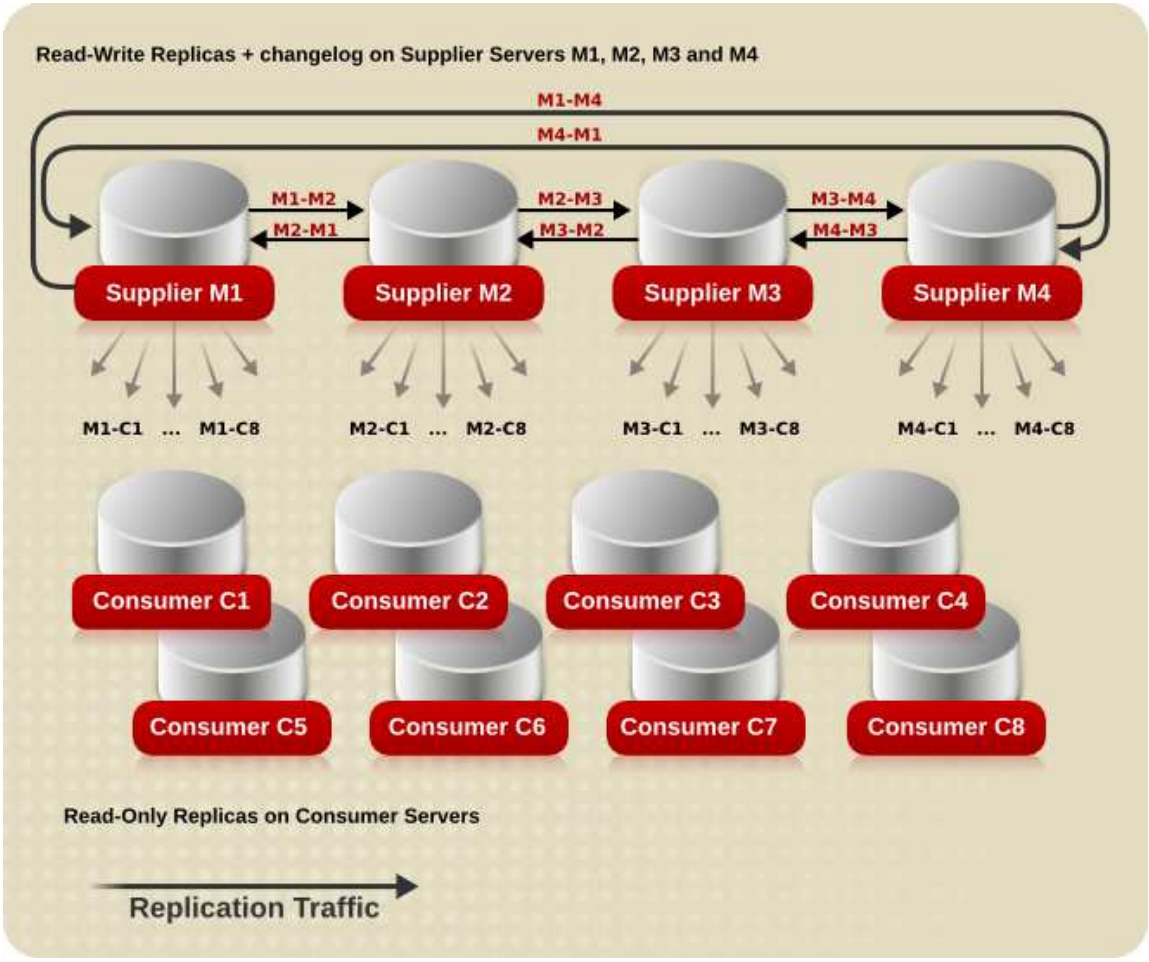


Figure 11.3. Multi-Master Replication (Four Masters)

Multi-master configurations have the following advantages:

- Automatic write failover when one supplier is inaccessible.
- Updates are made on a local supplier in a geographically distributed environment.

NOTE

The speed that replication proceeds depends on the speed of the network. Plan changes and directory configuration accordingly, and realize that changes to one directory may not be quickly replicated to other directories over slow links, such as wide-area networks, in geographically-distributed environments.

For the procedure to set up multi-master replication, see Section 11.5, “Configuring Multi-Master Replication”.

11.2.3. Cascading Replication

In a cascading replication scenario, one server, a *hub*, acts both as a consumer and a supplier. It holds a read-only replica and maintains a changelog, so it receives updates from the supplier server that holds the master copy of the data and, in turn, supplies those updates to the consumer. Cascading replication is very useful for balancing heavy traffic loads or to keep master servers based locally in geographically-distributed environments.

Figure 11.4, “Cascading Replication” shows an example of a simple cascading replication scenario, though it is possible to create more complex scenarios with several hub servers.

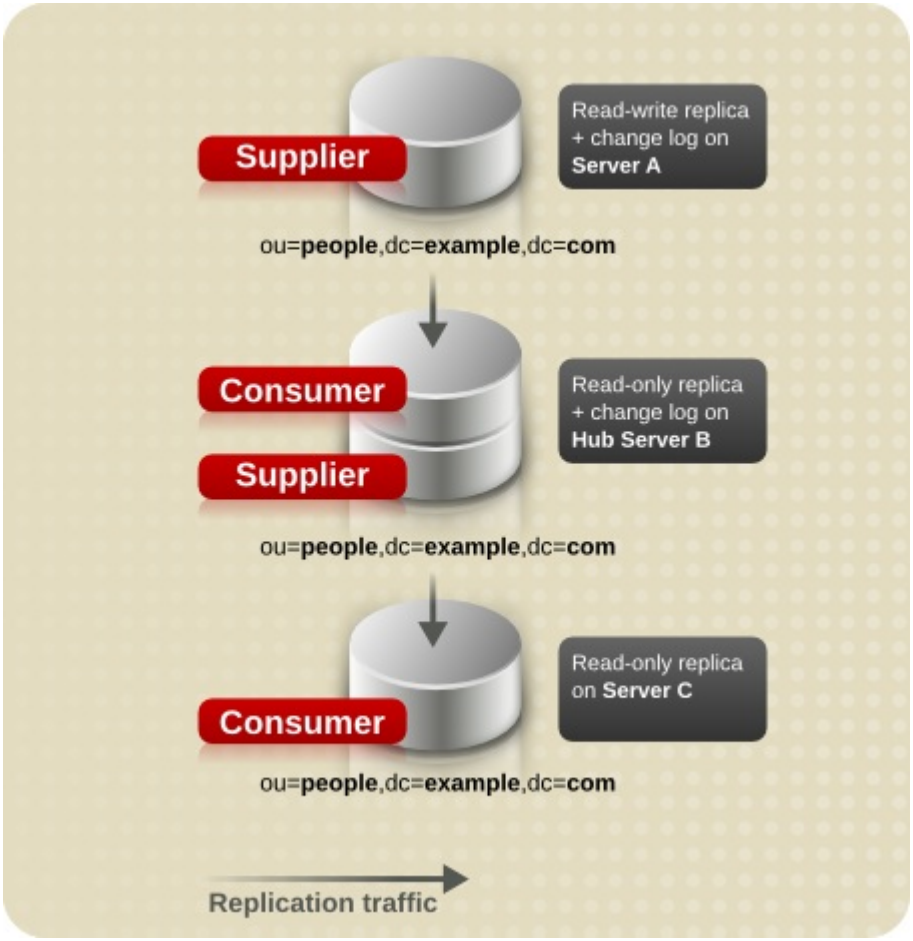


Figure 11.4. Cascading Replication

For information on setting up cascading replication, see Section 11.6, "Configuring Cascading Replication".

NOTE

Multi-master and cascading replication can be combined. For example, in the multi-master scenario illustrated in Figure 11.2, "Multi-Master Replication (Two Masters)", Server C and Server D could be hub servers that would replicate to any number of consumer servers.

11.3. CREATING THE SUPPLIER BIND DN ENTRY

A critical part of setting up replication is to create the entry, called the replication manager or supplier bind DN entry, that the suppliers use to bind to the consumer servers to perform replication updates.

The supplier bind DN must meet the following criteria:

- It must be unique.
- It must be created on the consumer server (or hub) and *not* on the supplier server.
- It must correspond to an actual entry on the consumer server.
- It must be created on *every* server that receives updates from another server.
- It must not be part of the replicated database for security reasons.
- It must be defined in the replication agreement on the supplier server.
- It must have an idle timeout period set to a high enough limit to allow the initialization process for large databases to complete. Using the ***nsIdleTimeOut*** operational attribute allows the replication manager entry to override the global ***nsslapd-idletimeout*** setting.

For example, the entry **cn=Replication Manager,cn=config** can be created under the **cn=config** tree on the consumer server. This would be the supplier bind DN that all supplier servers would use to bind to the consumer to perform replication operations.

NOTE

Avoid creating simple entries under the **cn=config** entry in the **dse.ldif** file. The **cn=cn=config** entry in the simple, flat **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, and particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will suffer. However, although Red Hat recommends not storing simple user entries under **cn=config** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **cn=config** since this centralizes configuration information.

On each server that acts as a consumer in replication agreements, create a special entry that the supplier will use to bind to the consumers. Make sure to create the entry with the attributes required by the authentication method specified in the replication agreement.

1. Stop the Directory Server. If the server is not stopped, the changes to the **dse.ldif** file will not be saved. See Section 1.3, "Starting and Stopping Servers" for more information on stopping the server.
2. Create a new entry, such as **cn=replication manager,cn=config**, in the **dse.ldif** file.
3. Specify a ***userPassword*** attribute-value pair.
4. Set an ***nsIdleTimeout*** period that gives the replication user a long enough time limit to allow replication initialization on large databases to complete.
5. If password expiration policy is enabled or ever will be enabled, disable it on the replication manager entry to prevent replication from failing due to passwords expiring. To disable the password expiration policy on the ***userPassword*** attribute, add the ***passwordExpirationTime*** attribute with a value of **20380119031407Z**, which means that the password will never expire.
6. Restart the Directory Server. See Section 1.3, "Starting and Stopping Servers" for more information on starting the server.

The final entry should resemble Example 11.1, "Example Supplier Bind DN Entry" .

Example 11.1. Example Supplier Bind DN Entry

```
dn: cn=replication manager,cn=config
objectClass: inetorgperson
objectClass: person
objectClass: top
cn: replication manager
sn: RM
userPassword: password
passwordExpirationTime: 20380119031407Z
nsIdleTimeout: 0
```

When configuring a replica as a consumer, use the DN of this entry to define the supplier bind DN.

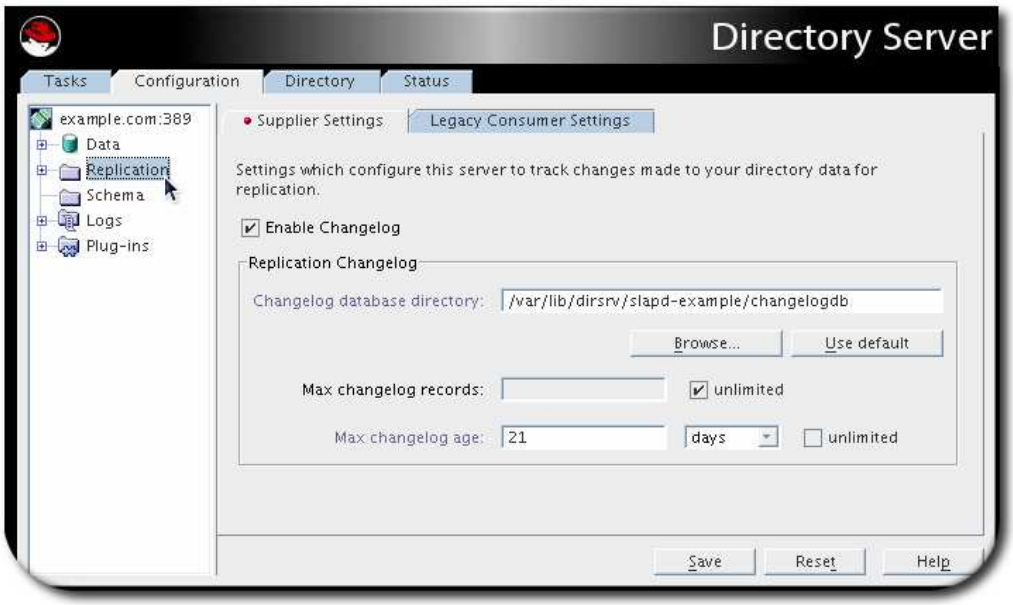
11.4. CONFIGURING SINGLE-MASTER REPLICATION

To set up single-master replication such as the configuration shown in Figure 11.1, "Single-Master Replication", between supplier Server A, which holds a read-write replica, and the two consumers Server B and Server C, which each hold a read-only replica, there are three major steps:

- Section 11.4.1, "Configuring the Read-Write Replica on the Supplier Server"
- Section 11.4.2, "Configuring the Read-Only Replica on the Consumer"
- Section 11.4.3, "Creating the Replication Agreement"

11.4.1. Configuring the Read-Write Replica on the Supplier Server

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



4. Check the **Enable Changelog** check box.

   This activates all of the fields in the pane below that were previously grayed out.

5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

6. Set the changelog parameters for the number and age of the log files.

   Clear the unlimited check boxes to specify different values.

7. Click **Save**.

2. Specify the replication settings required for a read-write replica.

   1. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

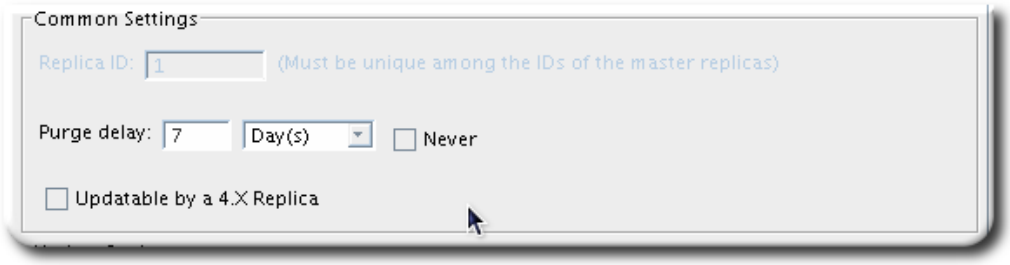
      The **Replica Settings** tab opens in the right-hand side of the window.

   2. Check the **Enable Replica** check box.

   3. In the **Replica Role** section, select the **Single Master** radio button.

   4. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.

      The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.



5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   The purge delay is how often the state information stored for the replicated entries is deleted.

6. Click **Save**.

11.4.2. Configuring the Read-Only Replica on the Consumer

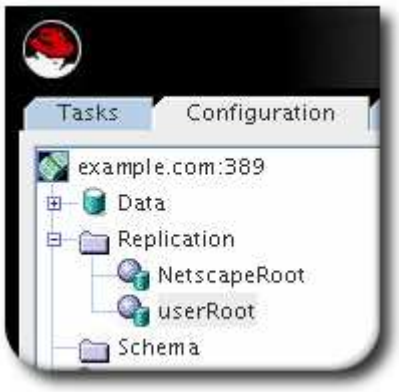
1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 11.3, "Creating the Supplier Bind DN Entry".

3. Specify the replication settings required for a read-only replica.

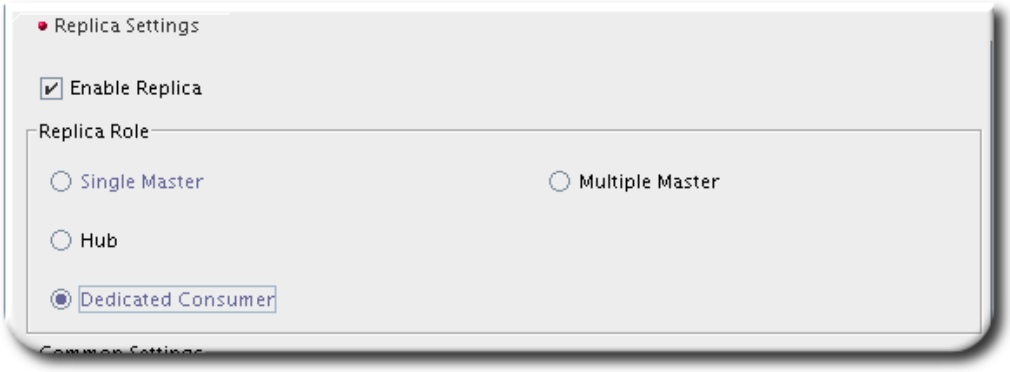
   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.



   The **Replica Settings** tab for that database opens in the right-hand side of the window.

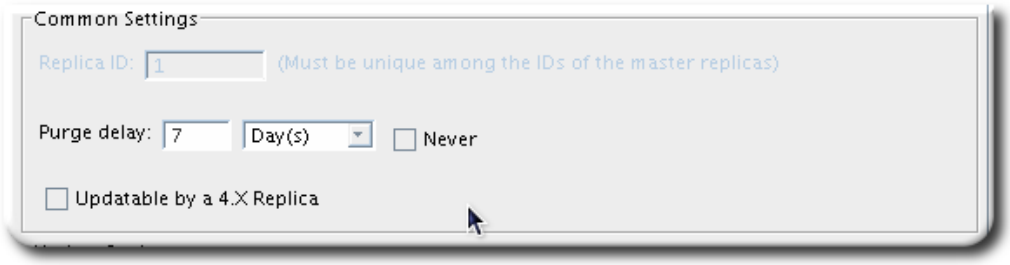
   3. Check the **Enable Replica** check box.



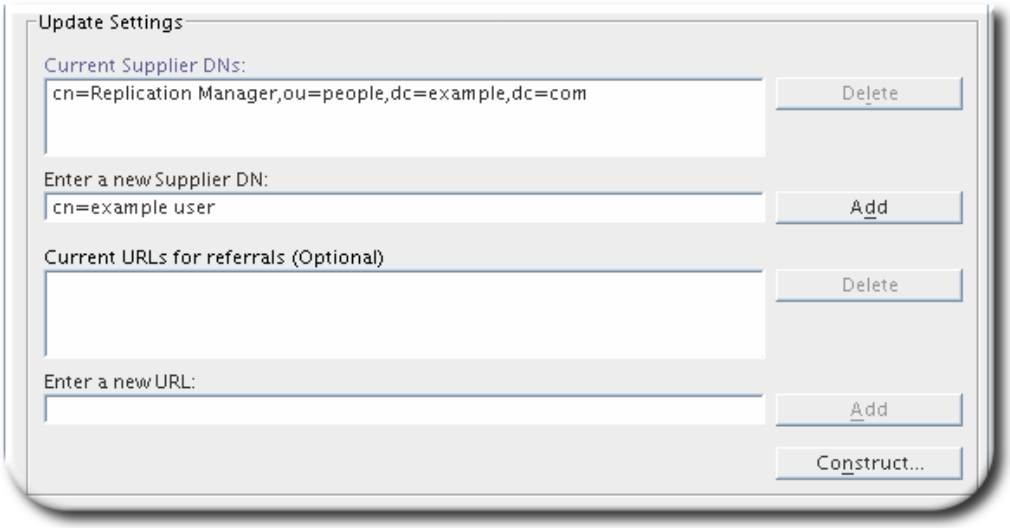
4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   This option indicates how often the state information stored for the replicated entries is purged.



6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control.

NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using SSL, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.

NOTE

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

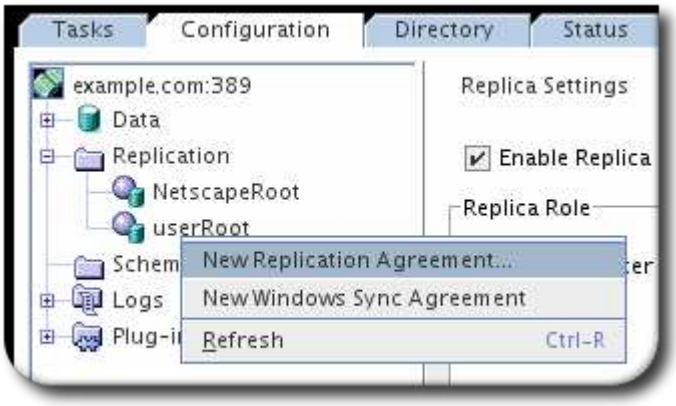
4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration.

11.4.3. Creating the Replication Agreement

Create one replication agreement for each read-only replica. For example, in the scenario illustrated in Figure 11.1, "Single-Master Replication", Server A has two replication agreements, one for Server B and one for Server C.

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.



Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.

3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.

- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.

- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

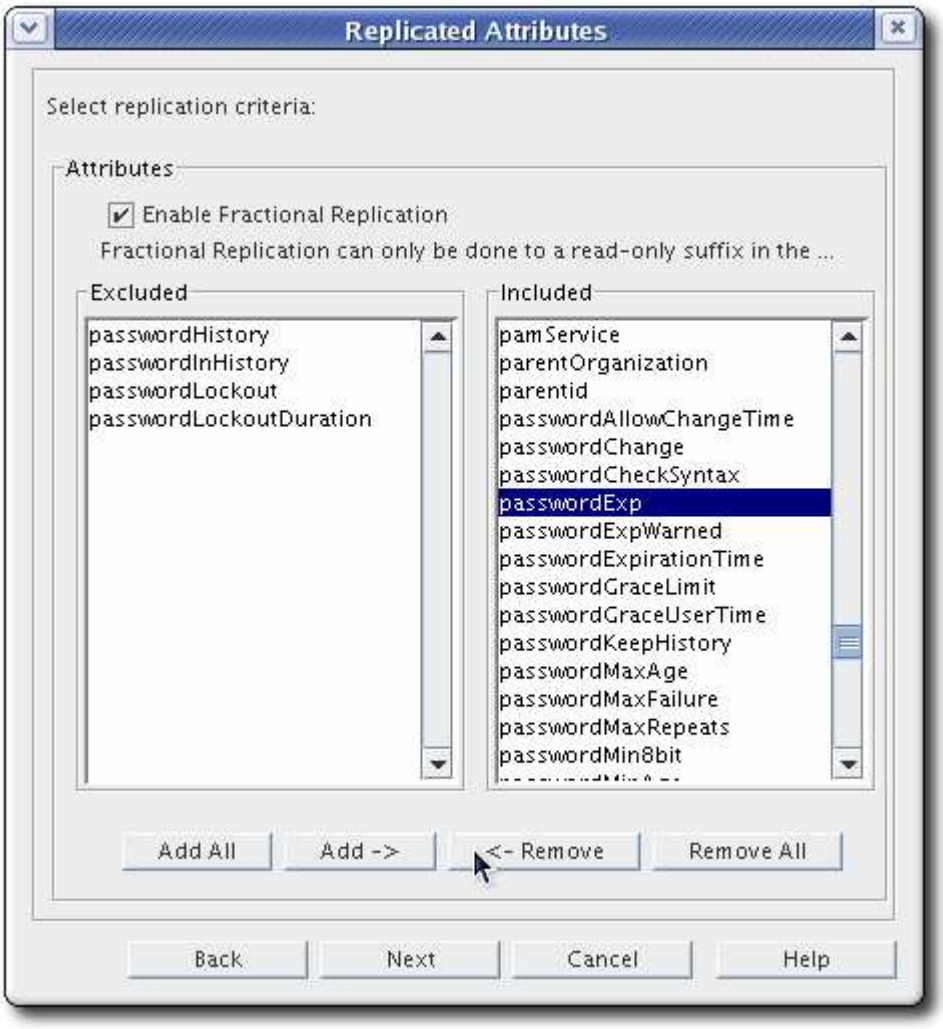
4. Select the connection type. There are three options:
   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS/SSL.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

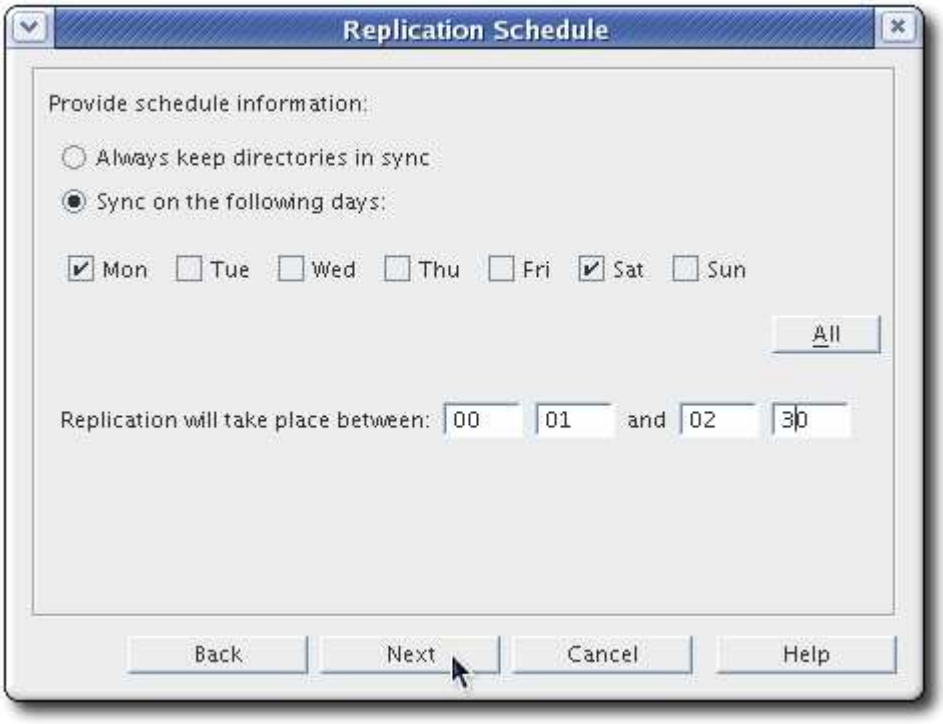
If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.
   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).
   - *Server TLS/SSL Certificate* uses the supplier's SSL certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

     Configuring SSL and certificate mapping is described in Section 7.4, "Setting up TLS/SSL".
   - *SASL/DIGEST-MD5*, like simple authentication, requires only the bind DN and password to authenticate. This can run over a standard or SSL/TLS connection.
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 7.12.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 7.11.3.1, "Configuring SASL Identity Mapping from the Console").
6. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



7. Set the schedule for when replication runs. By default, replication runs continually.

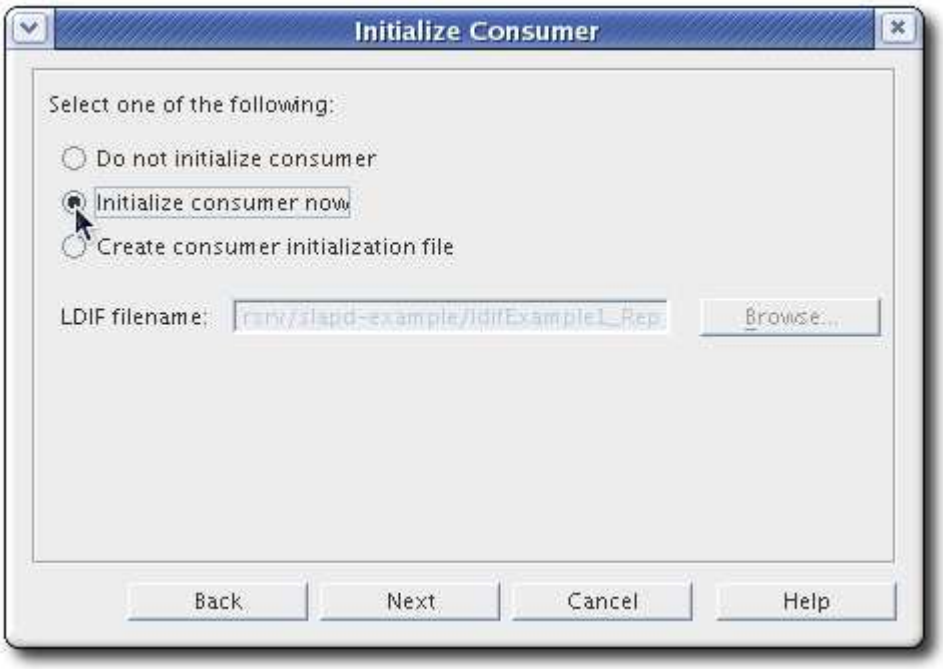


NOTE

The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

8. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see Section 11.15, "Initializing Consumers".

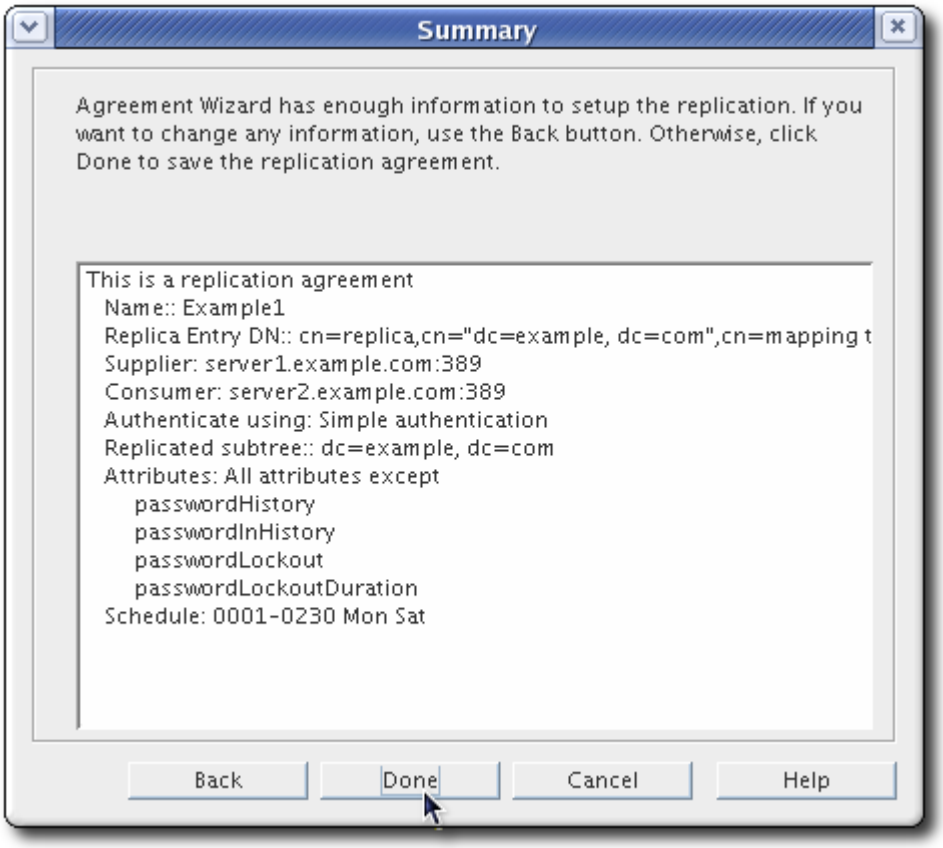


NOTE

Replication *will not* begin until the consumer is initialized.

Hit **Next**.

9. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.



The replication agreement is set up.

NOTE

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

11.5. CONFIGURING MULTI-MASTER REPLICATION

In a multi-master configuration, many suppliers can accept updates, synchronize with each other, and update all consumers. The consumers can send referrals for updates to all masters.

Directory Server supports 20-way multi-master replication, meaning that there can be up to 20 masters (and an unlimited number of hub suppliers) in a single replication scenario. Directory Server allows an unlimited number of consumers.

To set up multi-master replication, set up all of the consumers first, then set up the suppliers, and last, initialize all of the databases.

- Section 11.5.1, “Configuring the Read-Write Replicas on the Supplier Servers”
- Section 11.5.2, “Configuring the Read-Only Replicas on the Consumer Servers”
- Section 11.5.3, “Setting up the Replication Agreements”
- Section 11.5.4, “Preventing Monopolization of the Consumer in Multi-Master Replication”

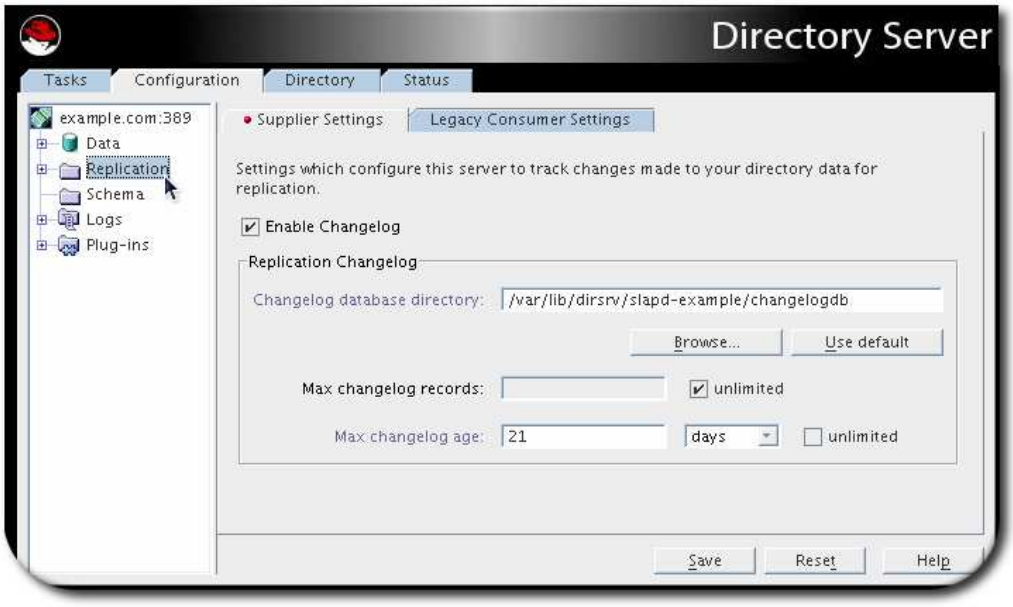
NOTE

More than 10 databases running with replication or more than on a supplier can cause performance degradation. To support that many consumers, introduce hub replicas between the suppliers and consumers. See Section 11.6, “Configuring Cascading Replication”.

11.5.1. Configuring the Read-Write Replicas on the Supplier Servers

Set up each supplier server. The first supplier configured should be used to initialize the other suppliers in the multi-master replication environment.

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



    4. Check the **Enable Changelog** check box.

       This activates all of the fields in the pane below that were previously grayed out.

    5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

    6. Set the changelog parameters for the number and age of the log files.

       Clear the unlimited check boxes to specify different values.

    7. Click **Save**.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. This is the special entry that the other suppliers will use to bind to this supplier, as in other supplier-consumer relationships. This is described in Section 11.3, “Creating the Supplier Bind DN Entry” .

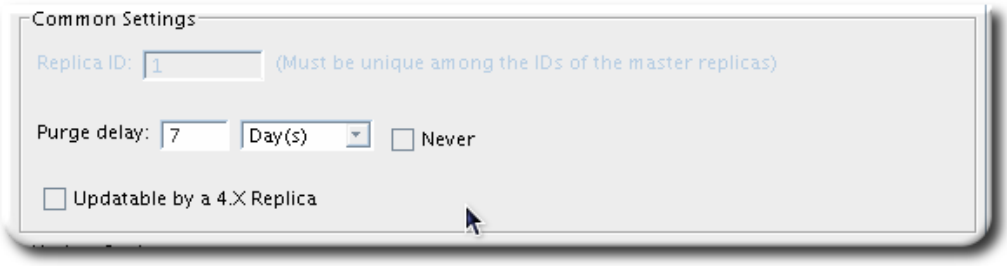
NOTE

For multi-master replication, it is necessary to create this supplier bind DN on the supplier servers as well as the consumers because the suppliers act as both consumer and supplier to the other supplier servers.

3. Specify the replication settings for the multi-mastered read-write replica.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

       The **Replica Settings** tab for that database opens in the right-hand side of the window.

3. Check the **Enable Replica** check box.
4. In the **Replica Role** section, select the **Multiple Master** radio button.
5. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.

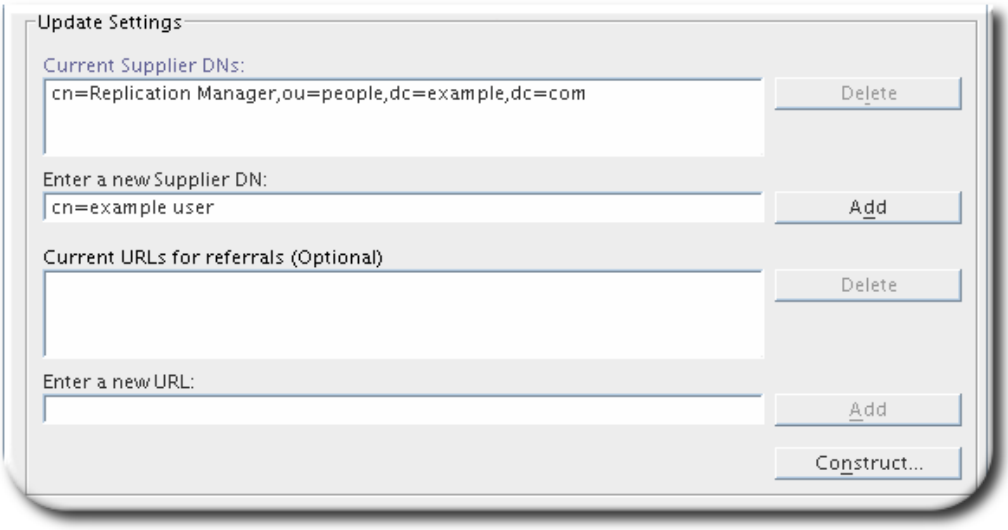


The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

6. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   The purge delay is how often the state information stored for the replicated entries is deleted.

7. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

8. Specify the LDAP URL (*ldap://hostname:port* or *ldap://IP_address:port*, with IPv4 or IPv6 addresses) for any supplier servers to which to refer updates, such as the other suppliers in the multi-master replication set. Only specify the URL for the supplier server.

   For clients to bind using SSL, specify a URL beginning with **ldaps://**.

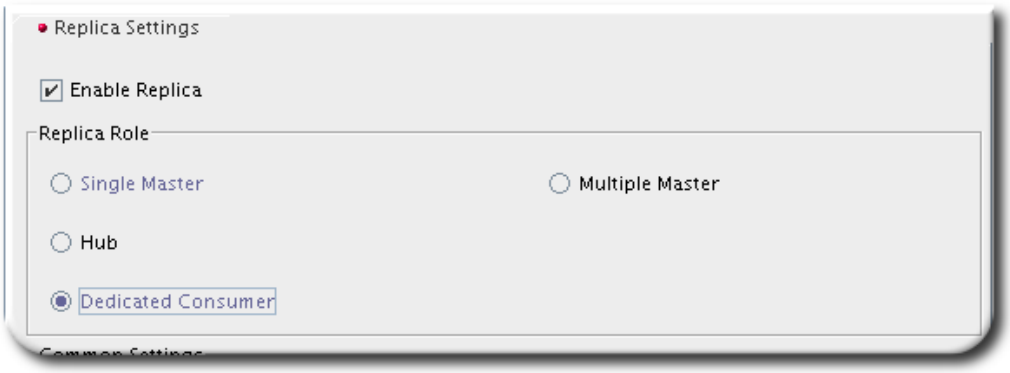
9. Click **Save**.

11.5.2. Configuring the Read-Only Replicas on the Consumer Servers

First, configure *every* consumer before creating any replication agreements.

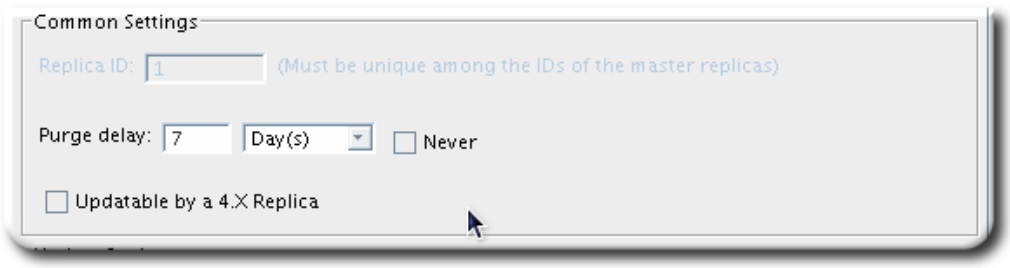
1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 11.3, "Creating the Supplier Bind DN Entry".
3. Specify the replication settings required for a read-only replica.
   1. In the Directory Server Console, select the **Configuration** tab.
   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

      The **Replica Settings** tab for that database opens in the right-hand side of the window.

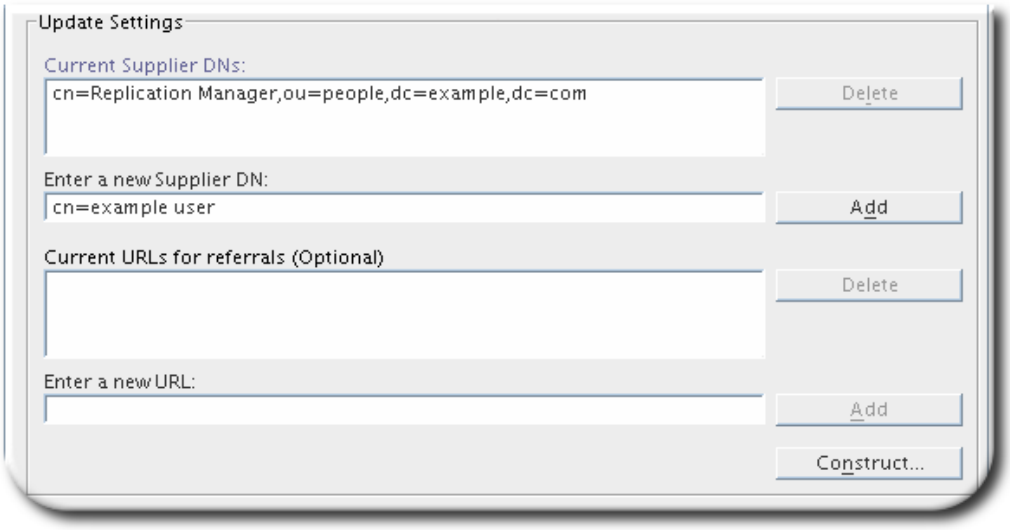


   3. Check the **Enable Replica** check box.
   4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.
   5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

      This option indicates how often the state information stored for the replicated entries is purged.



   6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using SSL, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.

NOTE

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

4. Click **Save**.

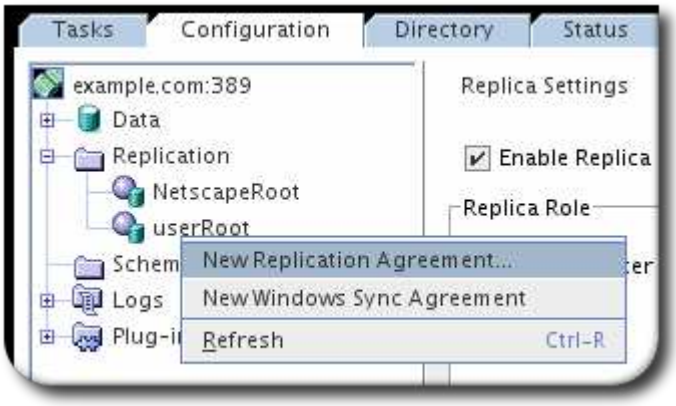
Repeat these steps for every consumer server in the replication configuration.

11.5.3. Setting up the Replication Agreements

NOTE

1. First set up replication agreements on a single supplier, the data master, between the other multi-master suppliers, and initialize all of the other suppliers.
2. Then create replication agreements for all other suppliers in the multi-master replication set, but *do not* reinitialize any of the suppliers.
3. Then create replication agreements for all of the consumers from the single data master, and initialize the consumers.
4. Then create replication agreements for all of the consumers from for all of the other suppliers, but *do not* reinitialize any of the consumers.

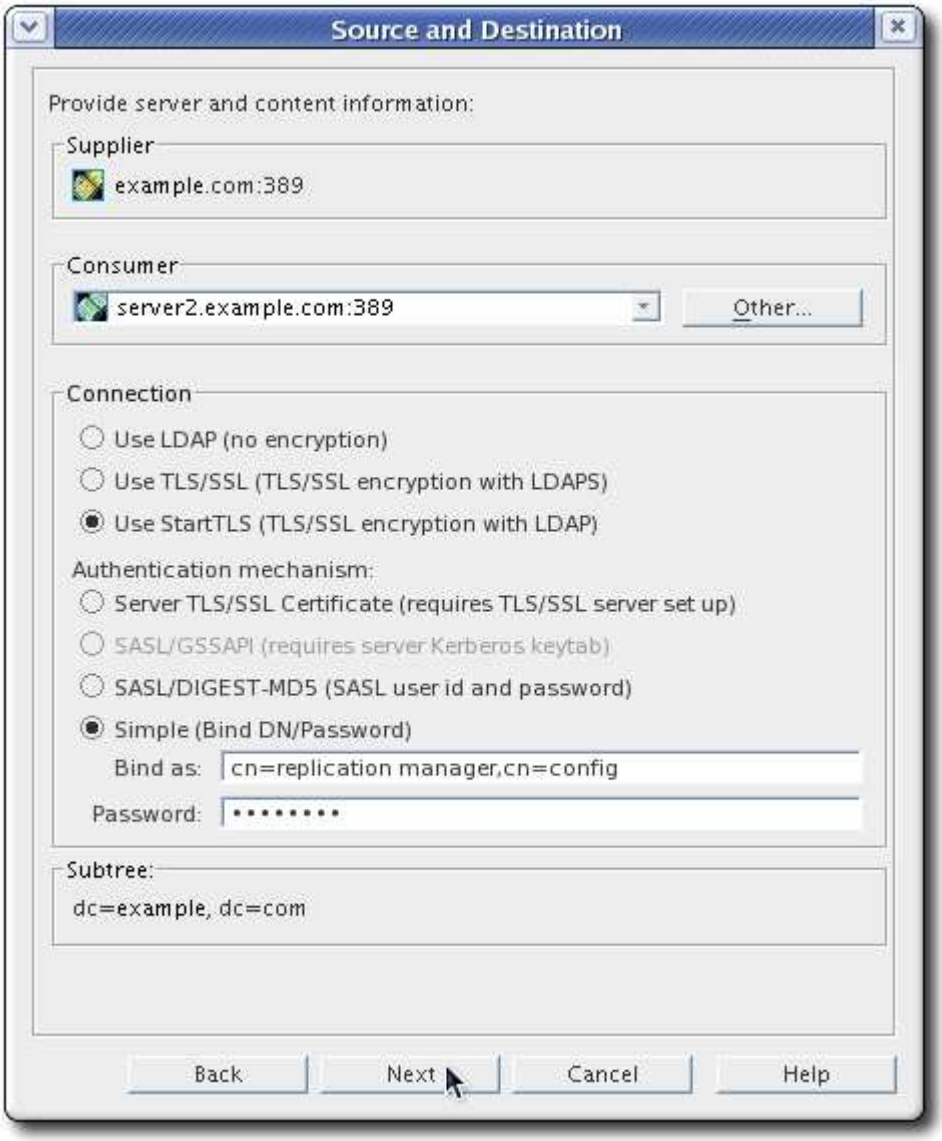
1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.



Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.

3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.



- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu. The server URL can be entered manually, in the format *hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses.

- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE

If attribute encryption is enabled, a secure connection is required for the encrypted attributes to be replicated.

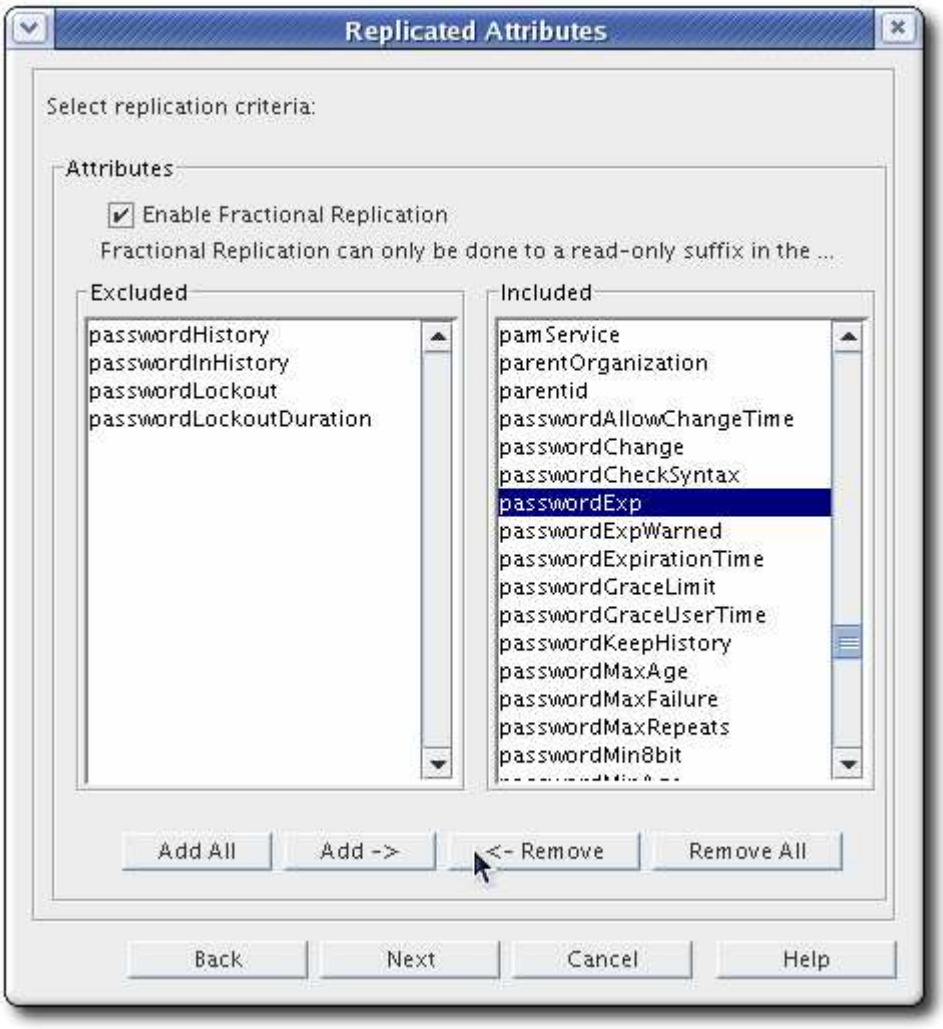
4. Select the connection type. There are three options:
   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS/SSL.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

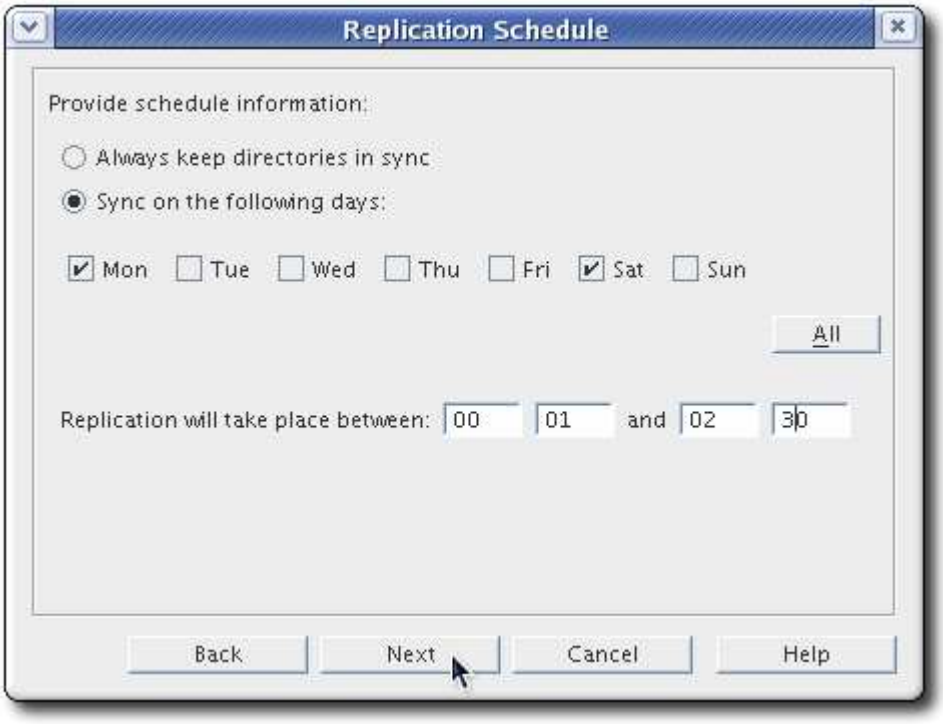
If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.
   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).
   - *Server TLS/SSL Certificate* uses the supplier's SSL certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

     Configuring SSL and certificate mapping is described in Section 7.4, "Setting up TLS/SSL".
   - *SASL/DIGEST-MD5*, like simple authentication, requires only the bind DN and password to authenticate. This can run over a standard or SSL/TLS connection.
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 7.12.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 7.11.3.1, "Configuring SASL Identity Mapping from the Console").
6. Hit **Next**.
7. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



8. Set the schedule for when replication runs. By default, replication runs continually.



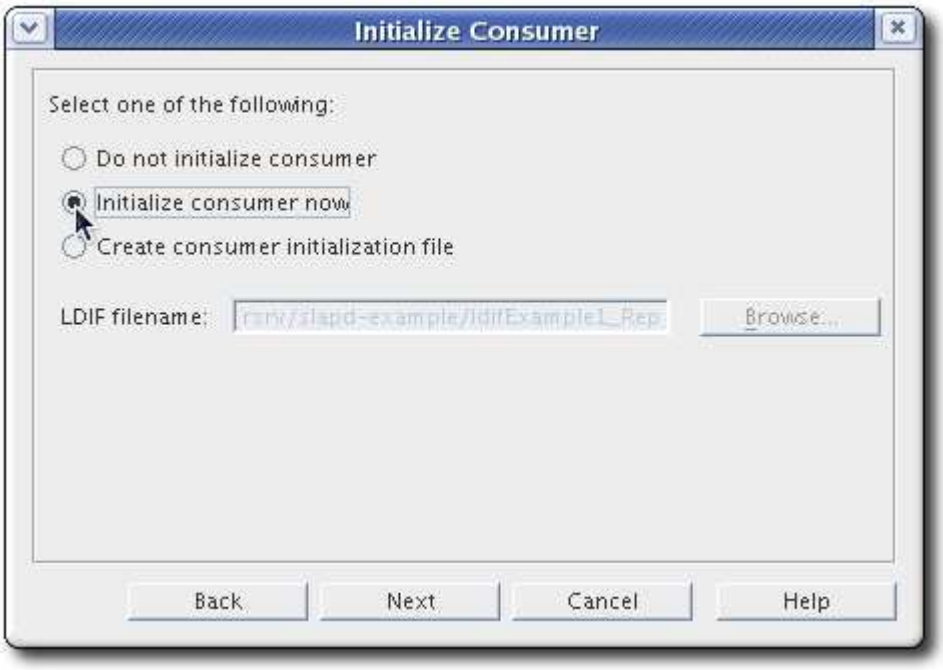
NOTE

The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

9. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see Section 11.15, "Initializing Consumers". For multi-master replication, consider the following:

   - Ensure one supplier has the complete set of data to replicate to the other suppliers. Use this one supplier to initialize the replica on all other suppliers in the multi-master replication set.
   - Initialize the replicas on the consumer servers from any of the multi-master suppliers.
   - Do not try to *reinitialize* the servers when the replication agreements are set up. For example, do not initialize server1 from server2 if server2 has already been initialized from server1. In this case, select **Do not initialize consumer**.



NOTE

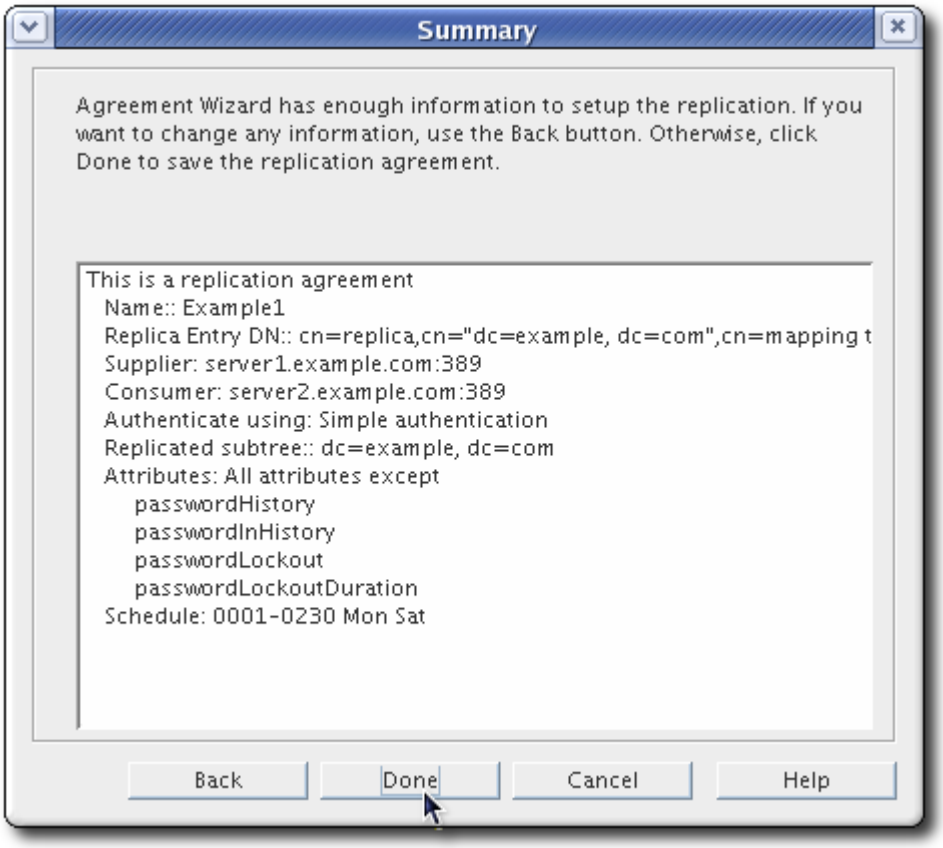
Replication *will not* begin until the consumer is initialized.

IMPORTANT

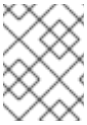
For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Hit **Next**.

10. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.



The replication agreement is set up.



NOTE

At the end of this procedure, all supplier servers will have mutual replication agreements, which means that they can accept updates from each other.

NOTE

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

11.5.4. Preventing Monopolization of the Consumer in Multi-Master Replication

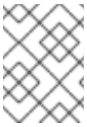
One of the features of multi-master replication is that a supplier acquires exclusive access to the consumer for the replicated area. During this time, other suppliers are locked out of direct contact with the consumer. If a supplier attempts to acquire access while locked out, the consumer sends back a busy response, and the supplier sleeps for several seconds before making another attempt. Normally, this is all right; the supplier simply sends its update to another consumer while the first consumer is locked and then send updates when the first consumer is free again.

A problem can arise if the locking supplier is under a heavy update load or has a lot of pending updates in the changelog. If the locking supplier finishes sending updates and then has more pending changes to send, it will immediately attempt to reacquire the consumer and will most likely succeed, since the other suppliers usually will be sleeping. This can cause a single supplier to monopolize a consumer for several hours or longer.

Two attributes address this issue, ***nsds5ReplicaBusyWaitTime*** and ***nsds5ReplicaSessionPauseTime***.

- *nsds5ReplicaBusyWaitTime*. The ***nsds5ReplicaBusyWaitTime*** attribute sets the amount of time in seconds a supplier should wait after a consumer sends back a busy response before making another attempt to acquire access. The default is **3** seconds.
- *nsds5ReplicaSessionPauseTime*. The ***nsds5ReplicaSessionPauseTime*** attribute sets the amount of time in seconds a supplier should wait between update sessions. Set this interval so that it is at least one second longer than the interval specified for ***nsds5ReplicaBusyWaitTime***. Increase the interval as needed until there is an acceptable distribution of consumer access among the suppliers. The default is **0**.

These two attributes may be present in the ***nsds5ReplicationAgreement*** object class, which is used to describe replication agreements. Set the attributes at any time by using **changetype:modify** with the **replace** operation. The change takes effect for the next update session if one is already in progress.



NOTE

If either attribute is set to a negative value, Directory Server sends the client a message and an **LDAP_UNWILLING_TO_PERFORM** error code.

The two attributes are designed so that the ***nsds5ReplicaSessionPauseTime*** interval will always be at least one second longer than the interval specified for ***nsds5ReplicaBusyWaitTime***. The longer interval gives waiting suppliers a better chance to gain consumer access before the previous supplier can re-access the consumer.

- If either attribute is specified but not both, ***nsds5ReplicaSessionPauseTime*** is set automatically to 1 second more than ***nsds5ReplicaBusyWaitTime***.
- If both attributes are specified, but ***nsds5ReplicaSessionPauseTime*** is less than or equal to ***nsds5ReplicaBusyWaitTime***, ***nsds5ReplicaSessionPauseTime*** is set automatically to 1 second more than ***nsds5ReplicaBusyWaitTime***.

If Directory Server has to automatically reset the value of ***nsds5ReplicaSessionPauseTime***, the value is changed internally only. The change is not visible to clients, and it not saved to the configuration file. From an external viewpoint, the attribute value appears as originally set.

Replica busy errors are not logged by default because they are usually benign. To see the errors, turn on the replication error logging, log level **8192**. The log levels are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference*.

11.6. CONFIGURING CASCADING REPLICATION

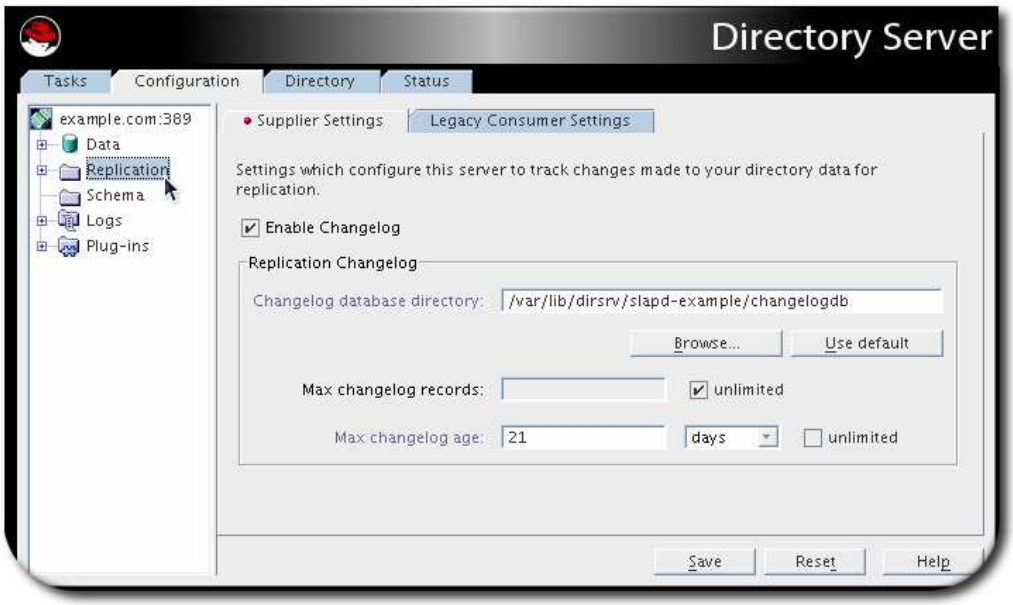
Setting up cascading replication, as shown in Figure 11.4, “Cascading Replication” , has three major steps, for each server in the scenario, the supplier on Server A, which holds a read-write replica; the consumer/supplier on hub Server B, which holds a read-only replica; and the consumer on Server C, which holds a read-only replica:

- Section 11.6.1, “Configuring the Read-Write Replica on the Supplier Server”
- Section 11.6.2, “Configuring the Read-Only Replica on the Consumer Server”
- Section 11.6.3, “Configuring the Read-Only Replica on the Hub”
- Section 11.6.4, “Setting up the Replication Agreements”

11.6.1. Configuring the Read-Write Replica on the Supplier Server

Next, configure the supplier server, which holds the original copy of the database:

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



4. Check the **Enable Changelog** check box.

   This activates all of the fields in the pane below that were previously grayed out.

5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

6. Set the changelog parameters for the number and age of the log files.

   Clear the unlimited check boxes to specify different values.

7. Click **Save**.

2. Specify the replication settings required for a read-write replica.

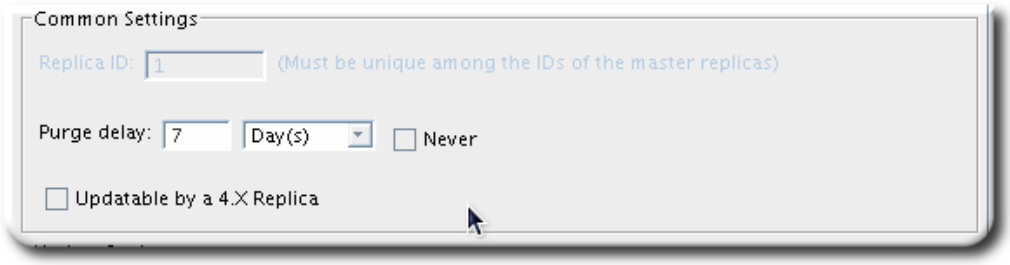
   1. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

      The **Replica Settings** tab opens in the right-hand side of the window.

   2. Check the **Enable Replica** check box.

   3. In the **Replica Role** section, select the **Single Master** radio button.

   4. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.



The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   The purge delay is how often the state information stored for the replicated entries is deleted.

6. Click **Save**.

After setting up the supplier replica, begin configuring the replication agreements.

11.6.2. Configuring the Read-Only Replica on the Consumer Server

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

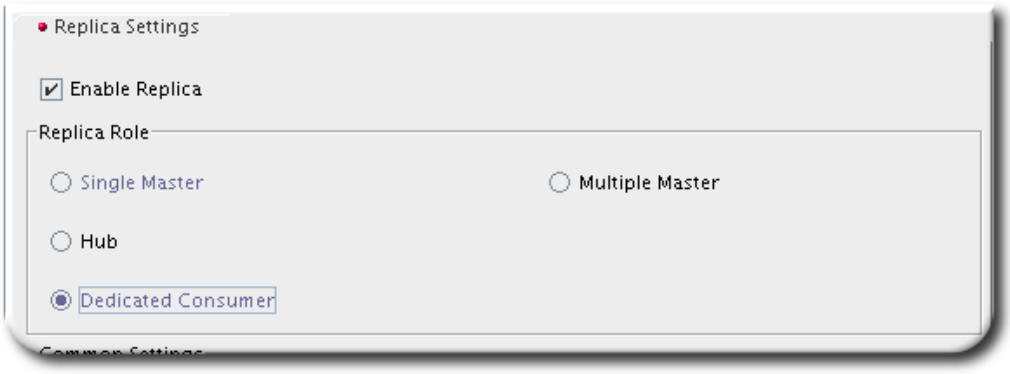
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 11.3, "Creating the Supplier Bind DN Entry".

3. Specify the replication settings required for a read-only replica.

   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

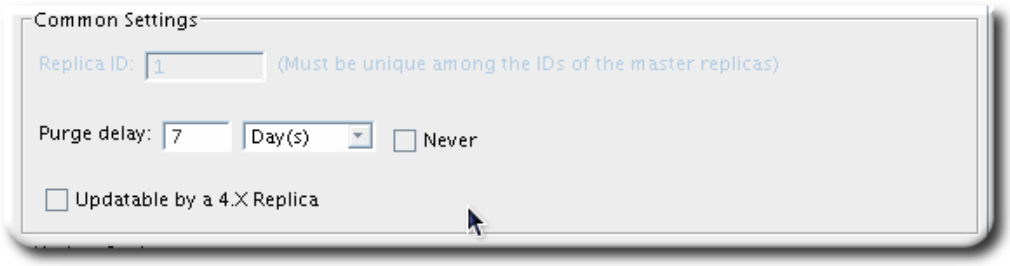
      The **Replica Settings** tab for that database opens in the right-hand side of the window.



   3. Check the **Enable Replica** check box.

   4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.

   5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.



      This option indicates how often the state information stored for the replicated entries is purged.

6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   In cascading replication, referrals are automatically sent to the hub server, which in turn refers the request to the original supplier. Therefore, set a referral to the original supplier to replace the automatically generated referral.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration, then configure the hub replica.

11.6.3. Configuring the Read-Only Replica on the Hub

Do this to set up a hub, which receives replication updates from the supplier and propagates them to consumers:

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 11.3, "Creating the Supplier Bind DN Entry".

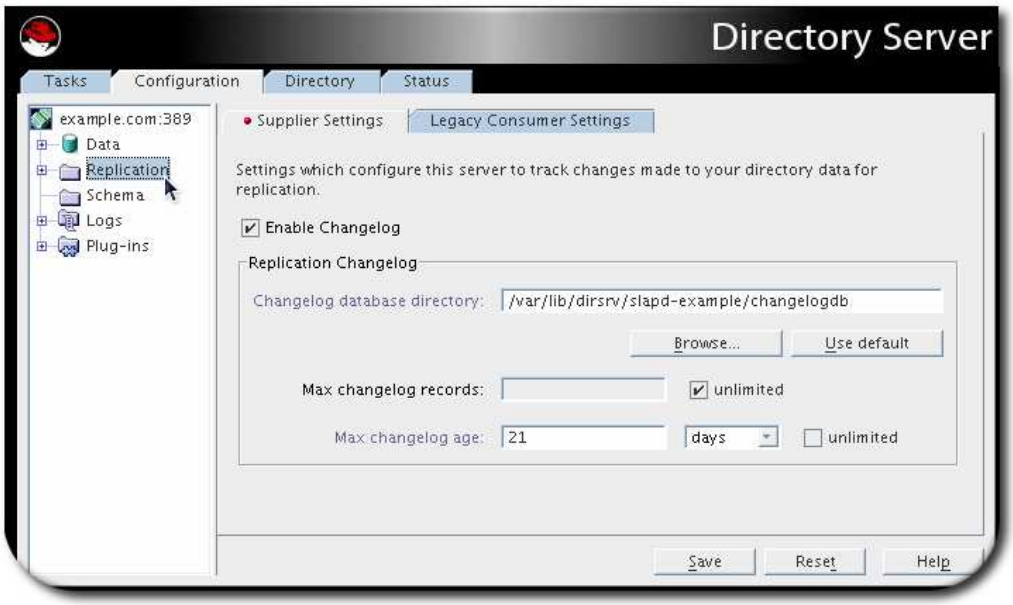
3. Create the changelog for the hub server.

   The hub must maintain a changelog even though it does not accept update operations because it records the changes sent from the supplier server.

   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, select the **Replication** folder.

   3. In the right-hand side of the window, select the **Supplier Settings** tab.



4. Check the **Enable Changelog** check box.

   This activates all of the fields in the pane below that were previously grayed out.

5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

6. Set the changelog parameters for the number and age of the log files.

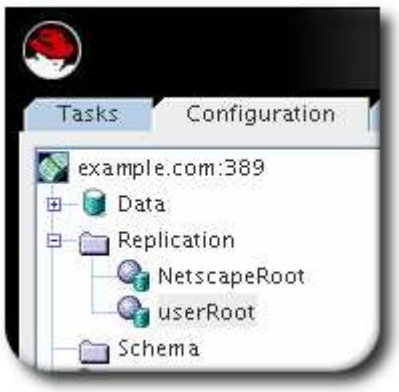
   Clear the unlimited check boxes to specify different values.

7. Click **Save**.

4. Specify the required hub replica settings.

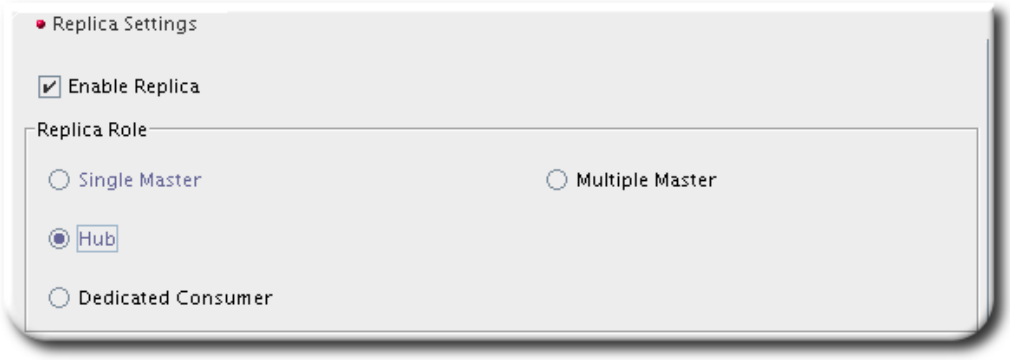
   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.



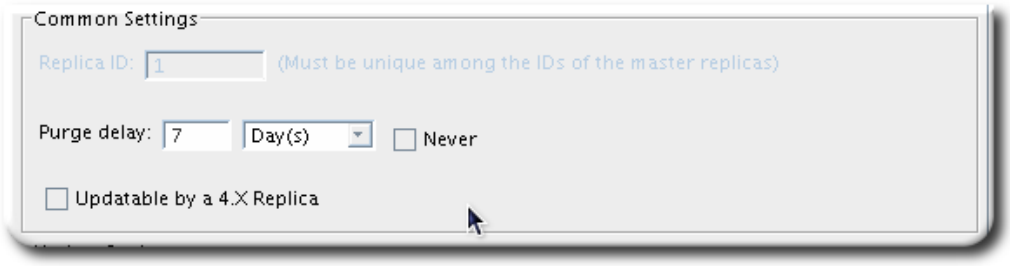
   The **Replica Settings** tab for that database opens in the right-hand side of the window.

   3. Check the **Enable Replica** check box.



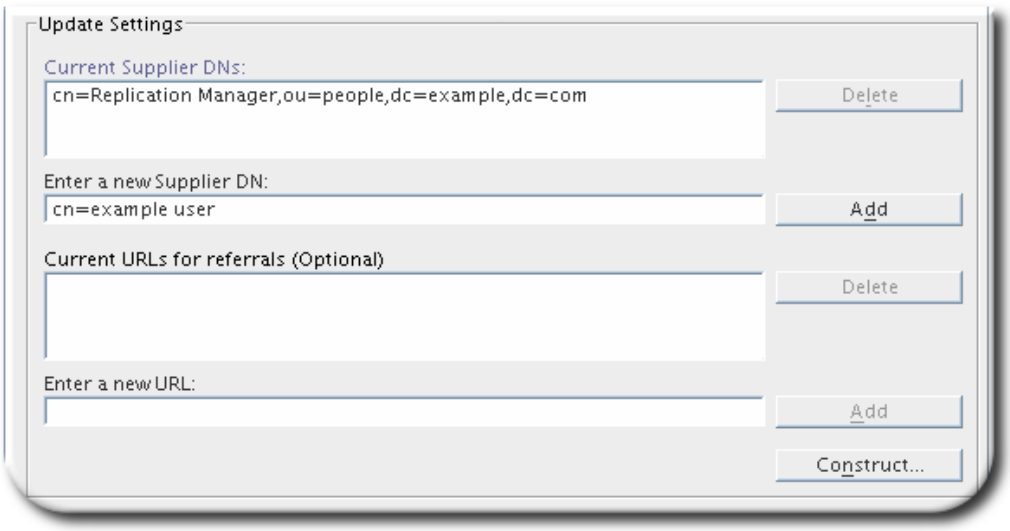
4. In the **Replica Role** section, select the **Hub** radio button.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.



This option sets how often the state information stored for the replicated entries is purged.

6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using SSL, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.

NOTE

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

5. Click **Save**.

When all the hubs are configured, then configure the supplier replica.

11.6.4. Setting up the Replication Agreements

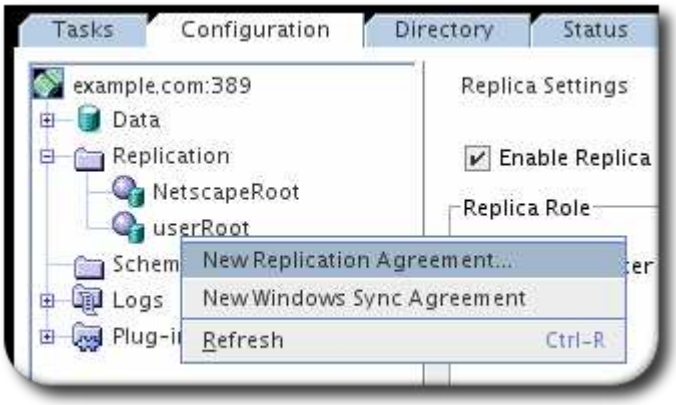
Cascading replication requires two sets of replication agreements, the first between the supplier and the hub and the second between the hub and the consumer. To set up the replication agreements:

1. Create the replication agreement on the supplier for the hub, then use the supplier server to initialize the replica on the hub server.

2. Then create the replication agreement on the hub for each consumer, and initialize the consumer replicas from the hub.

To set up a replication agreement:

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.



Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.

3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.

- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu. The server URL can be entered manually as *.hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses.
- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

4. Select the connection type. There are three options:
   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS/SSL.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.

   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).

   - *Server TLS/SSL Certificate* uses the supplier's SSL certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

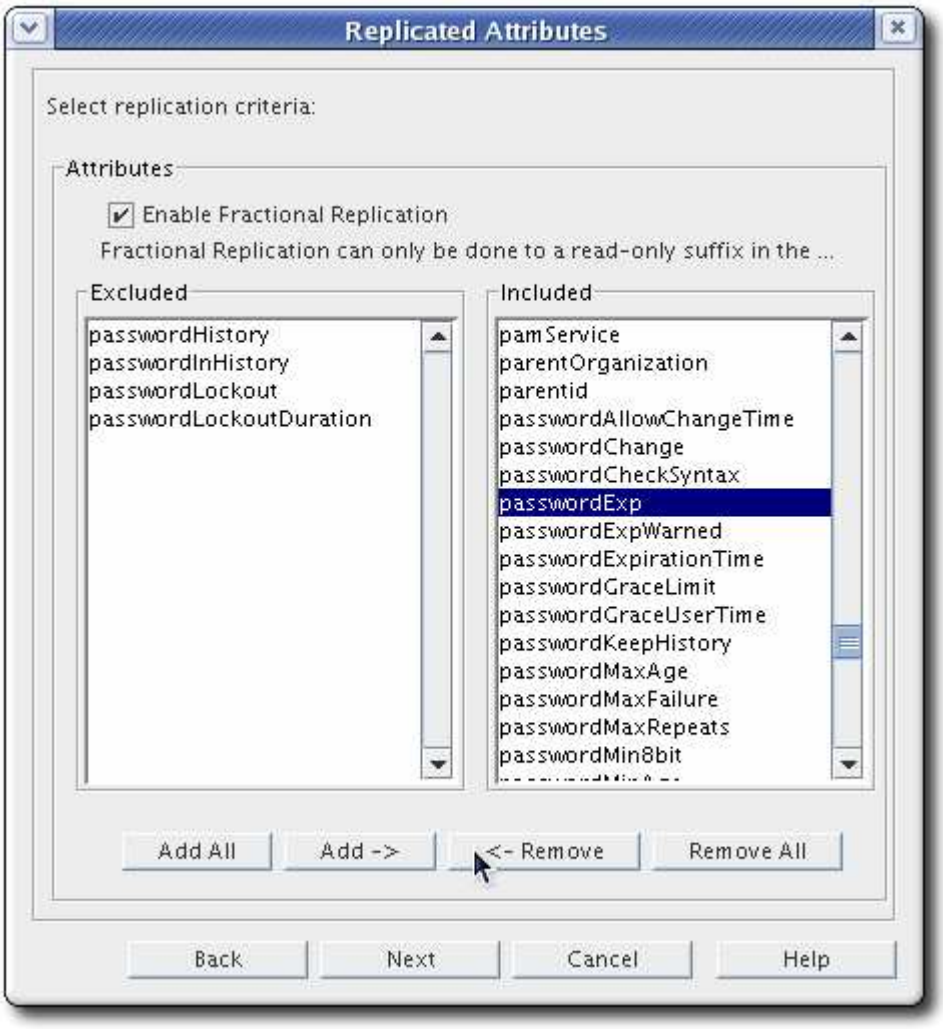
     Configuring SSL and certificate mapping is described in Section 7.4, "Setting up TLS/SSL".

   - *SASL/DIGEST-MD5*, like simple authentication, requires only the bind DN and password to authenticate. This can run over a standard or SSL/TLS connection.

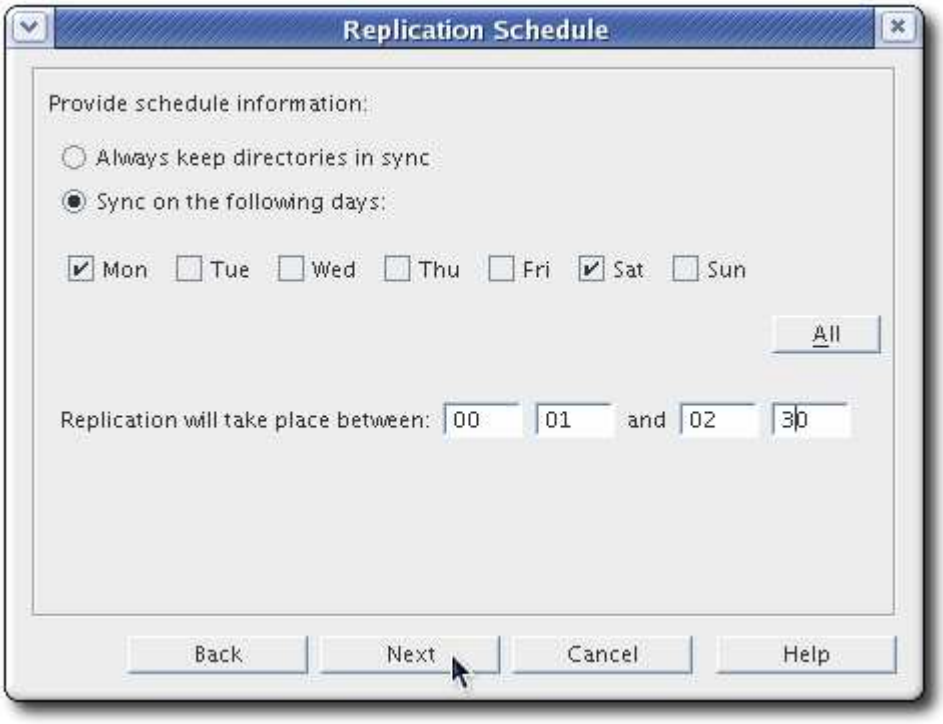
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 7.12.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 7.11.3.1, "Configuring SASL Identity Mapping from the Console").

6. Hit **Next**.

7. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



8. Set the schedule for when replication runs. By default, replication runs continually.



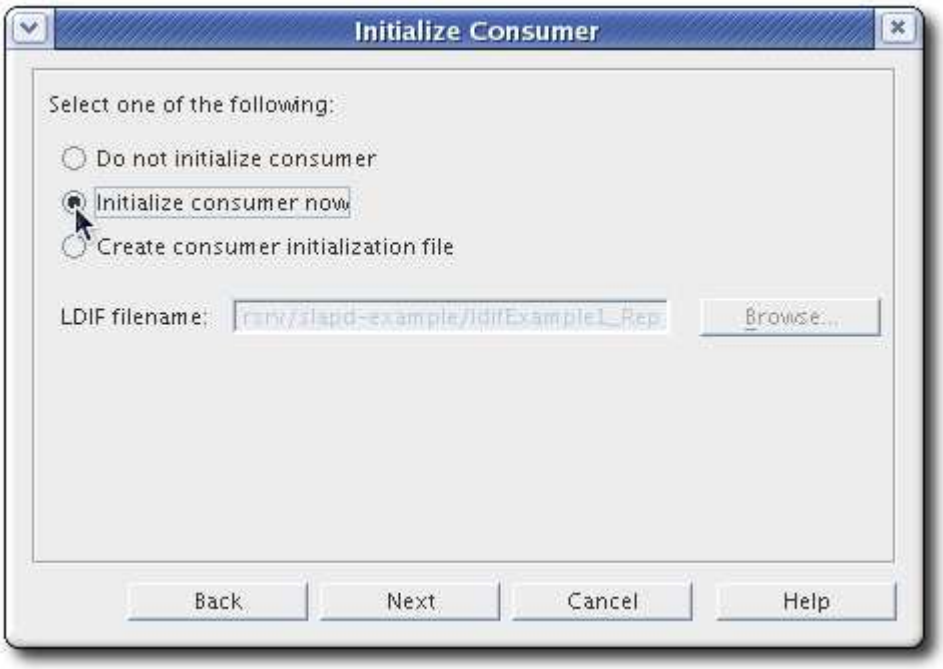
NOTE

The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

9. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see Section 11.15, "Initializing Consumers". For cascading replication, consider the following:

   - Create the supplier-hub replication agreement on the supplier first, and initialize the hub from the supplier.
   - Create the hub-consumer replication agreements on the hub, and initialize the consumers from the hub.



NOTE

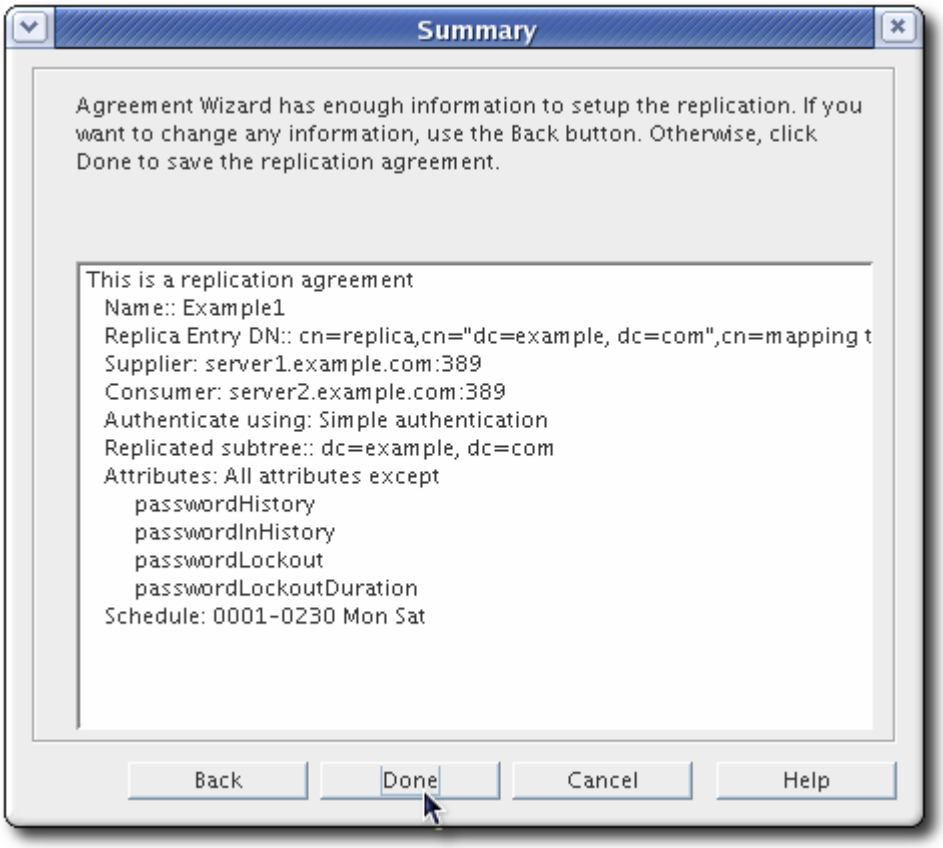
Replication *will not* begin until the consumer is initialized.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Hit **Next**.

10. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.



NOTE

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be change because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

11.7. CONFIGURING REPLICATION FROM THE COMMAND LINE

Replication can be configured on the command line by creating the appropriate replica and agreement entries on the servers. The process follows the same order as setting up replication through the Directory Server Console:

1. Create the supplier bind DN on every consumer, hub, and multi-master supplier (Section 11.3, "Creating the Supplier Bind DN Entry").
2. If the corresponding database and suffix do not exist on one of the replicas, create it (Section 2.1.1, "Creating Suffixes").
3. Configure the supplier replicas (Section 11.7.1, "Configuring Suppliers from the Command Line").
4. Configure consumers (Section 11.7.2, "Configuring Consumers from the Command Line").
5. Configure hubs for cascading replication (Section 11.7.3, "Configuring Hubs from the Command Line").
6. Create the replication agreements (Section 11.7.4, "Configuring Replication Agreements from the Command Line"). For cascading replication, create the agreement between the supplier and hub, then between the hub and consumers; for multi-master, create the agreements between all suppliers, then between the suppliers and consumers.
7. Lastly, initialize all of the consumers (Section 11.7.5, "Initializing Consumers Online from the Command Line"), if the consumers were not initialized when the replication agreement was created.

11.7.1. Configuring Suppliers from the Command Line

There are two steps to setting up the supplier replica. First, the changelog must be enabled, which allows the supplier to track changes to the Directory Server. Then, the supplier replica is created.

1. On the supplier server, use **ldapmodify** to create the changelog entry.

Example 11.2. Example Changelog Entry

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com -v -a

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb
nsslapd-changelogmaxage: 10d
```

There are two important attributes with the changelog.

- ***nsslapd-changelogdir*** sets the directory where the changelog is kept.
- ***nsslapd-changelogmaxage*** sets how long the changelog is kept; since the changelog can get very large, this helps trim the changelog to prevent affecting server performance and using up disk space. If this parameter is not set, the default is for the changelog to be kept forever.

The changelog entry attributes are described in Table 11.1, "Changelog Attributes". These attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference* .

2. Create the supplier replica.

Example 11.3. Example Supplier Replica Entry

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com -v -a

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 7
nsds5replicatype: 3
nsds5flags: 1
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=replication manager,cn=config
```

- *nsds5replicaroot* sets the subtree (suffix) which is being replicated.
- *nsds5replicatype* sets what kind of replica this database is. For either a single master or a multi-master supplier, this value must be **3**.
- *nsds5replicaid* sets the replica ID. The value must be unique among all suppliers and hubs; the valid range is **1** to **65534**.
- *nsds5ReplicaPurgeDelay* sets how long an entry holds its status information or how long a tombstone entry is kept before deleting the information. The default value is **604800** (one week).
- *nsds5flags* sets whether the replica writes to the changelog. For a supplier, this value must be **1**.

The replica entry attributes are described in Table 11.2, "Replica Attributes".

After creating every supplier which will take part in the replication setup, then begin creating the replication agreements.

Table 11.1. Changelog Attributes

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: extensibleObject	An object class which allows any other object class or attribute to be added to an entry.	

Object Class or Attribute	Description	Values
cn: changelog5	The naming attribute for the changelog entry.	Any string; the default usage is to set the common name to **changelog5**.
nsslapd-changelogdir: *directory*	Sets the file and directory changelog, to which the Directory Server writes changes.	Any directory; the default is **/var/lib/dirsrv/slapd-*instance_name*/changelogdb**.
nsslapd-changelogmaxage: *number unit*	Sets how long the changelog keeps an entry before purging it. This is not used by consumers, but is recommended for hubs and suppliers, which keep changelogs.	The *number* is any integer. The *unit* can be **s** for seconds, **m** for minutes, **h** for hours, **d** for days, or **w** for weeks.

Table 11.2. Replica Attributes

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: extensibleObject	An object class which allows any other object class or attribute to be added to an entry.	
objectclass: nsds5replica	An object class which allows replication attributes to be added to an entry.	
cn: replica	The naming attribute for the replica.	Any string; the default usage is to set the common name to **replica** for every configured replica.
nsds5replicaroot: *suffix*	Sets which subtree is replicated.	A root suffix associated with a database, since the entire database is replicated. For example: `dc=example,dc=com`
nsds5replicaid: *number*	The ID of the replica. This *must* be set to **65535** for consumers or hubs. For suppliers, this value must be a unique value.	**1** to **65534**, inclusive, for suppliers. **65535** for consumers and hubs.
nsds5replicatype: *number*	Sets the type of replica, either read-only or read-write.	**2** for consumers and hubs (read-only replicas) **3** for both single and multi-master suppliers (read-write replicas)
nsds5flags: *number*	Sets whether the replica writes to the changelog.	**0** means the replica does not write to the changelog; this is the default for consumers. **1** means the relics writes to the changelog; this is the default for hubs and suppliers.

Object Class or Attribute	Description	Values
nsds5ReplicaPurgeDelay: *number*	Sets the period of time in seconds to wait before purging the state information from an entry or purging tombstone entries. This setting is required for all types of replicas — suppliers, hubs, and consumers.	**0** (keep forever) to **2147483647** (the maximum 32-bit integer); the default value is **604800**, one week.
nsds5ReplicaBindDN: *DN*	The supplier bind DN used by the supplier to bind to the consumer. This is required for consumers, hubs, and multi-master suppliers, but not for single-master suppliers.	Any DN; the recommended DN is **cn=Replication Manager,cn=config**.  NOTE For security, it is strongly recommended that you do *not* use the Directory Manager as the supplier bind DN.
nsds5replicareferral: *URL*	*Optional.* An LDAP URL which a consumer or hub to which a consumer or hub can forward update requests. By default, update requests are sent to the masters for the consumer; use this parameter to override the default. The URL can use the format *ldap[s]://hostname:port* or *ldap[s]://IP_address:port*, with IPv4 or IPv6 addresses.	Any LDAP URL. For example: `nsds5replicareferral: ldap://supplier1.example.com:389`
nsDS5ReplicaReleaseTimeout: *seconds*	*Optional.* Sets the timeout in seconds after which a master releases a replica.	From **0** to the maximum 32-bit integer: **2147483647**. To disable the timeout, set the value to **0**. The default value is **60** seconds.

11.7.2. Configuring Consumers from the Command Line

On the consumer host, such as **consumer1.example.com**, create the replica entry. This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is. There are four key attributes:

- *nsds5replicaroot* sets the subtree (suffix) which is being replicated.
- *nsds5replicatype* sets what kind of replica this database is. For a consumer, this value must be **2**.
- *nsds5ReplicaBindDN* give the DN as which the supplier will bind to the consumer to make changes.
- *nsds5flags* sets whether the replica writes to the changelog. For a consumer, this value must be **0**.

This **ldapmodify** creates a new consumer replica on the **consumer1.example.com** host for the **dc=example,dc=com** subtree.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h consumer1.example.com -v -a

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 65535
nsds5replicatype: 2
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5flags: 0
```

The replica entry attributes are described in Table 11.2, "Replica Attributes". These attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference* .

11.7.3. Configuring Hubs from the Command Line

Hubs are intermediate read-only replicas which receive updates from suppliers and pass them on to other consumers. These are part of the cascading replication scenario, described in Section 11.2.3, "Cascading Replication".Creating the hub has two steps: first, creating the changelog database since the hub keeps a record of changes sent by the supplier, and second, configuring the hub replica.

1. On the hub server, such as **hub1.example.com**, use **ldapmodify** to create the changelog entry.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h hub1.example.com -v -a

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb
```

   There is one important attribute with the changelog, ***nsslapd-changelogdir***, which sets the directory where the changelog is kept.

   The changelog entry attributes are described in Table 11.1, "Changelog Attributes". These attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference* .

2. On the hub host, create the replica entry. This **ldapmodify** command creates a new hub replica on the **hub1.example.com** host for the **dc=example,dc=com** subtree.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h hub1.example.com -v -a

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaid: 65535
nsds5replicaroot: dc=example,dc=com
nsds5replicatype: 2
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5flags: 1
```

   This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is. There are five key attributes:

   - *nsds5replicaroot* sets the subtree (suffix) which is being replicated.
   - *nsds5replicatype* sets what kind of replica this database is. For a hub, this value must be **2**.
   - *nsds5replicaid* sets the ID of the replica. This *must* be set to **65535** for consumers or hubs.
   - *nsds5ReplicaPurgeDelay* sets how long an entry holds its status information or how long a tombstone entry is kept before deleting the information. The default value is **604800** (one week).
   - *nsds5ReplicaBindDN* give the DN as which the supplier will bind to the hub to make changes.
   - *nsds5flags* sets whether the replica writes to the changelog. For a hub, this value must be **1**.

The replica entry attributes are described in Table 11.2, "Replica Attributes". These attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference* .

11.7.4. Configuring Replication Agreements from the Command Line

When setting up replication agreements, first set them up between all suppliers, then between the suppliers and the hubs, and last between the hub and the consumers.

The replication agreement has to define eight separate attributes:

- The consumer host (***nsds5replicahost***) and port (***nsds5replicaport***).
- The DN for the supplier to use to bind with the consumer (***nsds5ReplicaBindDN***).
- The way that the supplier binds (***nsds5replicabindmethod***).
- Any credentials required (***nsDS5ReplicaCredentials***) for that bind method and specified DN.
- The subtree being replicated (***nsds5replicaroot***).
- The replication schedule (***nsds5replicaupdateschedule***).
- Any attributes which will *not* be replicated (***nsds5replicatedattributelist*** and ***nsDS5ReplicatedAttributeListTotal***).

Use **ldapmodify** to add a replication agreement to every supplier for every consumer which it will updated. For example:

Example 11.4. Example Replication Agreement Entry

```
dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
objectclass: top
objectclass: nsds5ReplicationAgreement
cn: ExampleAgreement
nsds5replicahost: consumer1
nsds5replicaport: 389
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5replicabindmethod: SIMPLE
nsds5replicaroot: dc=example,dc=com
description: agreement between supplier1 and consumer1
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList accountUnlockTime
memberof
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
nsds5replicacredentials: secret
nsds5BeginReplicaRefresh: start
```

The replication agreement attributes are listed in Table 11.3, "Replication Agreement Attributes". These attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference* .

After creating every replication agreement, begin initializing consumers.

Table 11.3. Replication Agreement Attributes

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: nsds5ReplicationAgreement	An operational object class which contains the replication agreement attributes.	
cn: *agreement_name*	The naming attribute for the replication agreement.	Any string.
nsds5replicahost: *hostname*	Gives the host name of the consumer server; the host name can be the fully qualified host and domain name. If TLS/SSL is enabled, the fully-qualified domain name is required. It is also possible to use IPv4 or IPv6 addresses instead of the host name.	Any host name. For example: `nsds5replicahost: consumer1`

Object Class or Attribute	Description	Values
nsds5replicaport: *number*	Gives the LDAP port for the consumer server. To use TLS/SSL, give the secure port number (**636** by default) and set the ***nsds5ReplicaTransportInfo*** attribute to **SSL**. To use Start TLS, which initiates a secure connection over a standard port, use the standard port, **389**, with the ***nsds5ReplicaTransportInfo*** attribute to **TLS**. To use simple authentication, use the standard port, **389**, with the ***nsds5ReplicaTransportInfo*** attribute to **LDAP**.	Any port number.
nsds5ReplicaTransportInfo: *method*	To use TLS/SSL, set this parameter to **SSL**. To use Start TLS, which initiates a secure connection over a standard port, set this parameter to **TLS**. To use simple authentication, set this parameter to **LDAP**.	empty or LDAP (standard connection over port 389) SSL (secure connection over the secure port, such as 636) TLS (secure connection over the standard port, 389, using Start TLS)
nsds5ReplicaBindDN: *DN*	The supplier bind DN used by the supplier to bind to the consumer. This is required for consumers, hubs, and multi-master suppliers, but not for single-master suppliers.	Any DN; the recommended DN is **cn=Replication Manager,cn=config**.

Object Class or Attribute	Description	Values
nsds5replicabindmethod: *type*	The connection type for replication between the servers. The connection type defines how the supplier authenticates to the consumer. Leaving the bind method empty or setting it to **SIMPLE** means that the server uses basic password-based authentication. This requires the ***nsds5ReplicaBindDN*** and ***nsds5ReplicaCredentials*** attributes to give the bind information. The **SSLCLIENTAUTH** option uses a secure connection. This requires setting the ***nsds5ReplicaTransportInfo*** attribute be set to **SSL** or **TLS**. For certificate-based authentication, the consumer server must also have a certificate mapping to map the subject DN in the supplier's certificate to the replication manager entry. Using **SASL/GSSAPI** requires that the supplier server have a Kerberos keytab (as in Section 7.12.2.2, "About the KDC Server and Keytabs"), and the consumer server have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 7.11.3.1, "Configuring SASL Identity Mapping from the Console"). The **SASL/DIGEST-MD5** setting, like **SIMPLE**, uses password-based authentication and requires the ***nsds5ReplicaBindDN*** and ***nsds5ReplicaCredentials*** attributes to give the bind information.  **NOTE** If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.	SIMPLE SSLCLIENTAUTH SASL/GSSAPI SASL/DIGEST-MD5

Object Class or Attribute	Description	Values
nsDS5ReplicaCredentials: *hash*	*Only for simple authentication.* Stores the hashed password used with the bind DN given for simple authentication.	
nsds5replicaroot: *suffix*	Sets which subtree is replicated.	A root suffix associated with a database, since the entire database is replicated. For example: `dc=example,dc=com`
description: *text*	A text description of the replication agreement.	Any text string. It is advisable to make this a useful description, such as *agreement between supplier1 and consumer1.*
nsds5replicatedattributelist: '(objectclass=*)' $ EXCLUDE *attributes*	*Optional.* Sets which attributes will *not* be replicated. The filter must be set to **"(objectclass=*)"**, and the list of attributes are separated by a single space. If this is the only fractional replication attribute set, then it applies to both incremental and total updates. If ***nsDS5ReplicatedAttributeListTotal*** is set, then this attribute applies to incremental updates.	`'(objectclass=*)' $ EXCLUDE authorityRevocationList memberof accountUnlockTime`
nsds5replicatedattributelisttotal: '(objectclass=*)' $ EXCLUDE *attributes*	*Optional.* Sets which attributes will *not* be replicated during a total update. The ***nsDS5ReplicatedAttributeList*** attribute must be set for incremental updates before these attributes can be set for total updates.	`'(objectclass=*)' $ EXCLUDE accountUnlockTime`

nsds5replicaupdateschedule: *start_time end_time days*	Sets the start and end time for the replication updates and the days on which replication occurs. If the schedule is omitted, replication will take place all the time.
	
	The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at**0001** on the next.
		Has the following value, with the start (*SSSS*) and end (*EEEE*) times set in the form **HHMM**
		
		The times are given in 24 hour clock format, so 0000 is midnight and 2359 is 11:59 PM. For example, the setting **1030 1630** schedules replication from 10:30 AM to 4:30 PM. The times cannot wrap around midnight, so the setting **2300 0100** is not valid.
		
		The days ranging from **0** (Sunday) to **6** (Saturday). Setting **06** schedules replication on Sunday and Saturday, while **135** schedules replication on Monday, Wednesday, and Friday.
		
		`nsds5replicaupdateschedule: SSSS EEEE DDDDDDD`
		
		For example, this schedules replication between midnight (**0000**) and 5am (**0500**) on Monday and Tuesday:
		
		`nsds5replicaupdateschedule: 0000 0500 12`

Object Class or Attribute	Description	Values
nsds5BeginReplicaRefresh: start	*Optional.* Performs an online (immediate) initialization of the consumer. If this is set, the attribute is only present as long as the consumer is being initialized; when the initialization is complete, the attribute is deleted automatically. If this is not set, then consumer initialization must be performed manually.	start To initialize the consumer, this attribute must have a value of **start**; any other value is ignored.

11.7.5. Initializing Consumers Online from the Command Line

An online initialization can be initiated from the command line by adding the ***nsds5replicarefresh*** attribute to the replication agreement entry. If the attribute is included when the replication agreement is created, initialization begins immediately. It can be added later to initialize the consumer at any time. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

   ```
   ldapsearch -x -h supplier1.example.com -p 389 -D "cn=directory manager" -W -s sub -b cn=config "
   (objectclass=nsds5ReplicationAgreement)"
   ```

   This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the ***nsds5BeginReplicaRefresh*** attribute:

   ```
   [root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com

   dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
   changetype: modify
   replace: nsds5BeginReplicaRefresh
   nsds5BeginReplicaRefresh: start
   ```

   ldapmodify does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the **ldapmodify** utility by hitting **Ctrl+C**.

When the initialization is complete, the ***nsds5BeginReplicaRefresh*** attribute is automatically deleted from the replication agreement entry.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Initializing consumers from the command line is also explained in Section 11.15.3, "Initializing Consumers Online Using the Command Line". Manually initializing consumers is explained in Section 11.15.4, "Manual Consumer Initialization Using the Command Line". The replication monitoring attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference*.

NOTE

For large databases, the ***nsslapd-idletimeout*** setting must be set to a large enough time period (or even an unlimited period) to allow the entire database to be initialized before the operation times out. Alternatively, the ***nsIdleTimeout*** setting for the supplier bind DN entry can be set high enough to allow the online initialization operation to complete, without having to change the global setting.

To keep data integrity, initialize the consumer databases from the appropriate supplier. Determining the correct supplier can be more difficult in mixed replication environments, but, even when manually initializing consumers, consider four things:

- Use one supplier, a *data master*, as the source for initializing consumers.
- Do not *reinitialize* a data master when the replication agreements are created. For example, do not initialize server1 from server2 if server2 has already been initialized from server1.
- For a multi-master scenario, initialize all of the other master servers in the configuration from one master.
- For cascading replication, initialize all of the hubs from a supplier, then initialize the consumers from the hubs.

11.8. TEMPORARILY SUSPENDING REPLICATION

By default, replication is enabled and active as soon as the replication agreement is created, and replication proceeds on the given schedule. There can be times when it is necessary to suspend replication, such as when a server is taken down for maintenance. An attribute can be added to the replication agreement entry which disables replication.

11.8.1. Disabling Replication

The ***nsds5ReplicaEnabled*** attribute on the replication entry sets whether the replication agreement is active. This attribute is missing by default and has a presumed value of **on**, meaning that replication is enabled.

To stop replication from a supplier, set the ***nsds5ReplicaEnabled*** attribute to **off**.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
add: nsds5ReplicaEnabled
nsds5ReplicaEnabled: off
```

11.8.2. Re-enabling Replication

1. Using **ldapmodify**, set the ***nsds5ReplicaEnabled*** attribute to **on**.

   ```
   [root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com

   dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
   changetype: modify
   add: nsds5ReplicaEnabled
   nsds5ReplicaEnabled: on
   ```

   Since the default value for this parameter is true, it is also possible to remove the attribute from the replication agreement entry.

   ```
   changetype: modify
   remove: nsds5ReplicaEnabled
   ```

2. Initiate an update, such as using the **replicate_now.sh** script as described in Section 11.16.2, “Forcing Replication Updates from the Command Line”.

11.9. MANAGING ATTRIBUTES WITHIN FRACTIONAL REPLICATION

As Section 11.1.7, “Replicating a Subset of Attributes with Fractional Replication” describes, fractional replication allows administrators to set attributes that are *excluded* from replication updates. Administrators can do this for a variety of performance reasons — to limit the number of large attributes that are sent over a network or to reduce the number of times that fixup tasks (like ***memberOf*** calculations) are run.

The list of attributes to exclude from replication are defined in the ***nsDS5ReplicatedAttributeList*** attribute. This attribute is part of the replication agreement and it can be configured in the replication agreement wizard in the Directory Server Console (or through the command line) when the replication agreement is created.

```
nsDS5ReplicatedAttributeList: (objectclass=*) $ EXCLUDE memberof authorityRevocationList accountUnlockTime
```

11.9.1. Setting Different Fractional Replication Attributes for Total and Incremental Updates

When fractional replication is first configured, the list of excluded attributes applies to every update operation. Meaning, this list of attributes is excluded for a total update as well as regular incremental updates. However, there can be times when attributes should be excluded from incremental updates for performance but should be included

in a total update to ensure the directory data sets are complete. In this case, it is possible to add a second attribute that defines a separate list of attributes to exclude from total updates, ***nsDS5ReplicatedAttributeListTotal***.

NOTE

nsDS5ReplicatedAttributeList is the primary fractional replication attribute. If only ***nsDS5ReplicatedAttributeList*** is set, then it applies to both incremental updates and total updates. If both ***nsDS5ReplicatedAttributeList*** and ***nsDS5ReplicatedAttributeListTotal*** are set, then ***nsDS5ReplicatedAttributeList*** only applies to incremental updates.

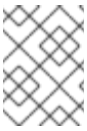
For example, every time a ***memberOf*** attribute is added to an entry, a memberOf fixup task is run to resolve the group membership. This can cause overhead on the server if that task is run every time replication occurs. Since a total update only occurs for a database which is newly-added to replication or that has been offline for a long time, running a memberOf fixup task after a total update is an acceptable option. In this case, the ***nsDS5ReplicatedAttributeList*** attribute lists ***memberOf*** so it is excluded from incremental updates, but ***nsDS5ReplicatedAttributeListTotal*** does not list ***memberOf*** so that it is included in total updates.

The exclusion list for incremental updates is set in the ***nsDS5ReplicatedAttributeList*** attribute for the replication agreement.

```
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList accountUnlockTime memberof
```

If ***nsDS5ReplicatedAttributeList*** is the only attribute set, then that list applies to both incremental and total updates. To set a separate list for total updates, add the ***nsDS5ReplicatedAttributeListTotal*** attribute to the replication agreement.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 389 -h
server.example.com -x

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
add: nsDS5ReplicatedAttributeListTotal
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
```



NOTE

The ***nsDS5ReplicatedAttributeList*** attribute must be set for incremental updates before ***nsDS5ReplicatedAttributeListTotal*** can be set for total updates.

11.9.2. Preventing "Empty" Updates from Fractional Replication

Fractional replication allows a list of attributes which are removed from replication updates (***nsDS5ReplicatedAttributeList***). However, a changed to an excluded attribute still triggers a modify event and generates an empty replication update.

The ***nsds5ReplicaStripAttrs*** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attribtes like ***modifiersName***.

For example, let's say that the ***accountUnlockTime*** attribute is excluded. John Smith's user account is locked and then the time period expires and it is automatically unlocked. Only the ***accountUnlockTime*** attribute has changed, and that attribute is excluded from replication. However, the operational attribute ***internalmodifytimestamp*** *also* changed. A replication event is triggered because John Smith's user account was modified – but the only data to send is the new modify time stamp and the update is otherwise emtpy. If there are a large number of attributes related to login times or password expiration times (for example), this could create a flood of empty replication updates that negatively affect server performance or that interfere with associated applications.

To prevent this, add the ***nsds5ReplicaStripAttrs*** attribute to the replication agreement to help tune the fractional replication behavior:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 389 -h
server.example.com -x

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
add: nsds5ReplicaStripAttrs
nsds5ReplicaStripAttrs: modifiersname modifytimestamp internalmodifiersname internalmodifytimestamp
```

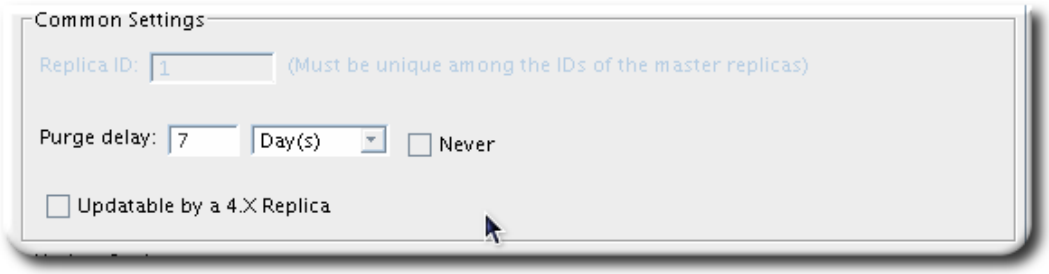
If a replication event is *not* empty, the stripped attributes *are* still replicated with the other changes. These attributes are removed from updates only if the event would otherwise be emtpy.

11.10. MAKING A READ-ONLY REPLICA UPDATABLE

Making a read-only server writable means changing the replica from a dedicated consumer or a hub to a supplier.

1. Make sure there are no updates in progress.
2. Stop the supplier server.
3. Open the Directory Server Console for the read-only replica.
4. In the **Configuration** tab, select **Replication**. In the right pane, select the **Enable changelog** check box.
5. Select the suffix, and, in the **Replica Settings** tab, change the replica role to a single master or multi-master.

6. Assign a unique replica ID.



7. Save the changes
8. Stop the Directory Server instance:

```
# systemctl stop dirsrv
```

9. Back up the **/etc/dirsrv/slapd-*instance*/dse.ldif** file:

```
# cp /etc/dirsrv/slapd-instance/dse.ldif /etc/dirsrv/slapd-instance/dse.ldif-1
```

Do not name the backup file **dse.ldif.bak**. Directory Server uses this file name to keep a known working copy of the **dse.ldif** file.

10. Edit the **/etc/dirsrv/slapd-*instance*/dse.ldif** file.
    a. Search for replication agreements. For example:

```
dn: cn=replica,cn=dc\5c3Dexample\5c2Cdc\5c3Dcom,cn=mapping tree,cn=config
```

    b. Remove the line containing the ***nsState*** attribute from every replication agreement.
11. Start the Directory Server instance:

```
# systemctl start dirsrv.target
```

12. Monitor the error log file for error messages. For details, see Section E.2.4, “Viewing Logs”.

    If the replication fails:

    a. Delete all replication agreements: Section 11.11, “Removing a Supplier from the Replication Topology” .

    b. Disable replication: Section 11.8.1, “Disabling Replication”.

    c. Remove the changelog configuration: Section 11.13, “Removing the Changelog”.

    d. Restart the Directory Server and Admin Console:

    ```
    # systemctl restart dirsrv.target
    # systemctl restart dirsrv-admin.service
    ```

    e. Enable replication: Section 11.8.2, “Re-enabling Replication”.

    f. Create replication agreements: Section 11.2, “Replication Scenarios”.

11.11. REMOVING A SUPPLIER FROM THE REPLICATION TOPOLOGY

Removing a supplier cleanly from the replication topology is more complex than simply removing the supplier entry. This is because every supplier in the topology stores information about other suppliers in the topology, and they retain that information even if a supplier is suddenly unavailable.

Information about the replication topology – all of the suppliers which are supplying updates to each other and other replicas within the same replication group – is contained in a set of metadata called the *replica update vector (RUV)*. The RUV contains information about the supplier like its ID and URL, its latest change state number for changes made on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

To remove a supplier cleanly, this metadata must be removed along with the configuration entries.

1. *On the replica to remove* , put the database into read-only mode to prevent any updates.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -p 389 -h dead-replica.example.com

dn: cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-readonly
nsslapd-readonly: on
```

   Allow a few minutes for the replica to flush all of its pending changes.

2. *On all other suppliers in the topology* , delete the replication agreement with the replica to be removed.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -p 389 -h replica1.example.com

dn: cn=Agmt_with_dead-replica,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: delete
```

3. *On the replica to remove* , get the replica ID for the replica to remove. This is in the ***nsds5replicaid*** attribute in the configuration entry.

```
ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b cn=config objectclass=nsds5replica
nsds5replicaid

dn: cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsds5replicaid: 55
...
```

4. *On the replica to remove* , remove all replication agreement entries and its own configuration entry.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -p 389 -h dead-replica.example.com

dn: cn=to_replica1,cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: delete
```

```
...

dn: cn=to_replica2,cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: delete

dn: cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: delete
```

5. *On one of the other servers in the topology* , run the **CLEAN*ALL*RUV** replication task. This operation will be propagated to all the servers in the replication environment. The task name has the format **CLEANALLRUV***replica#*. For example, for a supplier with the replica ID 55, the task name is **CLEANALLRUV55**.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
replace: nsds5task
nsds5task: CLEANALLRUV55
```

It is possible to monitor the progress of the task on the other replicas by searching on the tombstone entry on each replica:

```
ldapsearch -xLLL -D "cn=directory manager" -W -h remaining-replica.example.com -b
"dc=example,dc=com" '(&(nsuniqueid=ffffffff-ffffffff-ffffffff-ffffffff)(objectclass=nstombstone))' nsds50ruv
```

11.12. MANAGING DELETED ENTRIES WITH REPLICATION

When an entry is deleted, it is not immediately removed from the database. Rather, it is converted into a *tombstone* entry, a kind of backup entry that is used by servers in replication to resolve conflicts. The tombstone entry preserves state information about the original entry.

If there is ever a replication conflict, then the supplier uses the replica ID (the server where the change was initiated) and the timestamp of the change in the change sequence number to resolve the conflict. The oldest change wins.

As with deleted entries, deleted attributes are also kept in tombstone entries.

Tombstones are not preserved indefinitely. A purge job is run periodically, at a specified interval (set in the ***nsDS5ReplicaTombstonePurgeInterval*** attribute); the purge removes old tombstone entries. Tombstone entries are saved for a given amount of time (set in the ***nsDS5ReplicaPurgeDelay*** attribute); one a tombstone entry is older than the delay period, it is reaped at the next purge job.

Both the purge delay and the purge interval are set on the replica entry for a supplier server in the **cn=replica,cn=***replicated suffix***,cn=mapping tree,cn=config** configuration entry.

There are two considerations when defining the purge settings for replication:

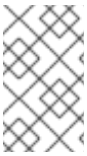
- The purge operation is time-consuming, especially if the server handles a lot of delete operations. Do not set the purge interval too low or it could consume too many server resources and affect performance.
- Suppliers use change information, including tombstone entries, to prime replication after initialization. There should be enough of a backlog of changes to effectively re-initialize consumers and to resolve replication conflicts. Do not set the purge delay (the age of tombstone entries) too low or you could lose information required to resolve replication conflicts.

  Set the purge delay so that it is slightly longer than the longest replication schedule in the replication topology. For example, if the longest replication interval is 24 hours, keep tombstone entries around for 25 hours. This ensures that there is enough change history to initialize consumers and prevent the data stored in different suppliers from diverging.

To change the purge settings:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x  -h supplier1.example.com

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
replace: nsDS5ReplicaTombstonePurgeInterval
nsDS5ReplicaTombstonePurgeInterval: 43200 # in seconds, 12 hours
```

```
changetype: modify
replace: nsDS5ReplicaPurgeDelay
nsDS5ReplicaPurgeDelay: 90000 # in seconds, 25 hours
```



NOTE

To clean up the tombstone entries and the state information immediately, set set a very small value to the ***nsDS5ReplicaTombstonePurgeInterval*** and ***nsDS5ReplicaPurgeDelay*** attributes. Both attributes have values set in seconds, so the purge operations can be initiated almost immediately.

WARNING

Always use the purge intervals to clean out tombstone entries from the changelog. **Never delete tombstone entries manually.**

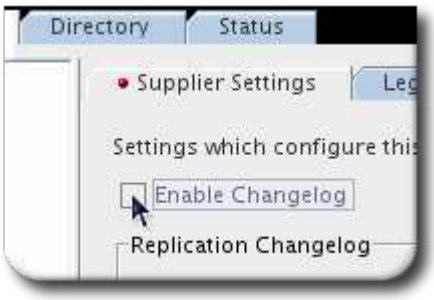
11.13. REMOVING THE CHANGELOG

The changelog is a record of all modifications on a given replica that the supplier uses to replay these modifications to replicas on consumer servers (or suppliers in the case of multi-master replication).

If a supplier server goes offline, it is important to be able to delete the changelog because it no longer holds a true record of all modifications and, as a result, should not be used as a basis for replication. A changelog can be effectively deleted by deleting the log file.

To remove the changelog from the supplier server:

1. In the Directory Server Console, select the **Configuration** tab.
2. Select the **Replication** folder in the left navigation tree and then the **Supplier Server Settings** tab in the right pane.
3. Clear the **Enable Changelog** check box.



4. Click **Save**.
5. Restart Directory Server. See Section 1.3.1, “Starting and Stopping Directory Server from the Console” .
6. Reinitialize the consumers. See Section 11.15, “Initializing Consumers”.

11.13.1. Moving the Changelog to a New Location

Sometimes, it may be necessary to delete the changelog while the supplier server is still running and continuing to log changes.

In this case, simply move the current changelog file to a new location. The server then automatically creates a new changelog in the specified directory.

After moving the changing, reinitialize the consumers; any changes to the changelog requires consumer reinitialization.

11.14. TRIMMING THE REPLICATION CHANGELOG

The Directory Server changelog manages a list of received and processed changes. It includes changes clients run on the server and additionally directory changes from other replication partners. Changelog entries are removed only when both of the following conditions meet:

- The entry was successfully transferred to all replicas.
- The entry exceeded the time set in the ***nsslapd-changelogmaxage*** parameter or the total number of records exceeded the value set in ***nsslapd-changelogmaxentries***. For further details, see the parameter descriptions in the *Red Hat Directory Server Configuration, Command, and File Reference* .

A record, including all subsequently created records, remains in the changelog until it is successfully replicated to all servers in the topology. This situation occurs, for example, if a Directory Server master was removed from the topology, but the replica update vector (RUV) has not been removed.

The file size of the changelog is not automatically reduced if you set a lower value in the ***nsslapd-changelogmaxage*** or ***nsslapd-changelogmaxentries*** parameter. To recreate the changelog:

1. Stop the Directory Server instance:

```
# service dirsrv stop instance_name
```

2. Remove the changelog files:

```
# rm /var/lib/dirsrv/slapd-instance_name/changelogdb/*
```

3. Start the instance:

```
# service dirsrv start instance_name
```

4. Reinitialize the instance from an incoming replica. See Section 11.15, “Initializing Consumers”.

11.15. INITIALIZING CONSUMERS

Once a replication agreement is created, the consumer must be *initialized*; that is, the data must be physically copied from the supplier server to the consumer servers.

NOTE

Replication *will not* begin until the consumer is initialized.

- Section 11.15.1, “When to Initialize a Consumer”
- Section 11.15.2, “Online Consumer Initialization Using the Console”
- Section 11.15.3, “Initializing Consumers Online Using the Command Line”
- Section 11.15.4, “Manual Consumer Initialization Using the Command Line”
- Section 11.15.5, “Filesystem Replica Initialization”

NOTE

For large databases, the ***nsslapd-idletimeout*** setting must be set to a large enough time period (or even an unlimited period) to allow the entire database to be initialized before the operation times out. Alternatively, the ***nsIdleTimeout*** setting for the supplier bind DN entry can be set high enough to allow the online initialization operation to complete, without having to change the global setting.

11.15.1. When to Initialize a Consumer

Consumer initialization involves copying data from the supplier server to the consumer server. Once the subtree has been physically placed on the consumer, the supplier server can begin replaying update operations to the consumer server.

Under normal operations, the consumer should not ever have to be reinitialized. However, any time there is a chance that there is a big discrepancy between the supplier's data and the consumer's, reinitialize the consumer. For example, if the data on the supplier server is restored from backup, then all consumers supplied by that server should be reinitialize. As another example, if the supplier has not been able to contact the consumer for a long time, like a week, the supplier may determine that the consumer is too far out of date to be updated, and must be reinitialized.

The consumer can either be initialized online using the Console or manually using the command line. Online consumer initialization using the Console is an effective method of initializing a small number of consumers. However, since each replica is initialized in sequence, this method is not suited to initializing a large number of replicas. Online consumer initialization is the method to use when the consumer is initialized as part of configuring the replication agreement on the supplier server.

Manual consumer initialization using the command line is a more effective method of initializing a large number of consumers from a single LDIF file.

11.15.2. Online Consumer Initialization Using the Console

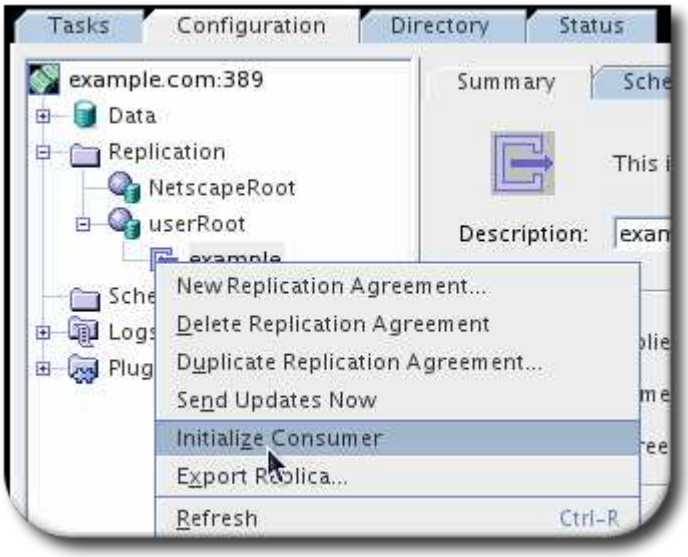
Online consumer initialization using the Console is the easiest way to initialize or reinitialize a consumer. However, for replicating across a slow link, this process can be very time-consuming, and manual consumer initialization using the command line may be a more efficient approach. This is described in more detail Section 11.15.4, "Manual Consumer Initialization Using the Command Line".

NOTE

When a consumer server is being initialized using the online consumer creation method, all operations (including searches) on the replica are referred to the supplier server until the initialization process is completed.

To initialize or reinitialize a consumer online:

1. Create a replication agreement.
2. On the supplier server, on the Directory Server Console, select the **Configuration** tab.
3. Expand the **Replication** folder, then expand the replicated database. Right-click the replication agreement, and choose **Initialize Consumer** from the pop-up menu.



A message opens warning that any information already stored in the replica on the consumer will be removed.

4. Click **Yes** in the confirmation box.

Online consumer initialization begins immediately. To check the status of the online consumer initialization, open the **Summary** tab in the **Status** box. If online consumer initialization is in progress, the status shows that a replica is being initialized.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

To update this window, right-click the replicated database icon in the navigation tree, and choose **Refresh Replication Agreements**. When online consumer initialization finishes, the status changes to reflect this.

For more information about monitoring replication and initialization status, see Section 11.22, "Monitoring Replication Status".

11.15.3. Initializing Consumers Online Using the Command Line

Online consumer initialization can be performed through the command line by adding the ***nsds5BeginReplicaRefresh*** attribute to the replication agreement entry. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

```
ldapsearch -h supplier1.example.com -p 389 -D "cn=directory manager" -W -s sub
    -b cn=config "(objectclass=nsds5ReplicationAgreement)"
```

   This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the ***nsds5BeginReplicaRefresh*** attribute:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
replace: nsds5BeginReplicaRefresh
nsds5BeginReplicaRefresh: start
```

   ldapmodify does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the **ldapmodify** utility by hitting **Ctrl+C**.

To check the initialization status, do an **ldapsearch** for the replication agreement entry.

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s base -b
'cn=ExampleAgreement,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config' '(objectclass=*)'
```

If the ***nsds5BeginReplicaRefresh*** attribute is present, the initialization is still in progress. If the initialization is complete, then the attribute ***nsds5ReplicaLastInitStatus*** shows the status. If the initialization was successful, the value of ***nsds5ReplicaLastInitStatus*** is **Total update succeeded**. If the initialization was not successful, this attribute shows information about the error; check the error logs for both the supplier and consumer for additional information.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

The replication monitoring attributes are described in more detail in the *Directory Server Configuration and Command-Line Tool Reference*.

11.15.4. Manual Consumer Initialization Using the Command Line

Manual consumer initialization using the command line is the fastest method of consumer initialization for sites that are replicating very large numbers of entries. However, the manual consumer initialization process is more complex than the online consumer initialization process. Red Hat suggests using the manual process whenever the online process is inappropriate due to performance concerns.

Initializing or reinitializing a server manually has three steps:

1. Create a replication agreement.
2. Export the replica on the supplier server to an LDIF file.

   See Section 11.15.4.1, "Exporting a Replica to LDIF" .
3. Import the LDIF file with the supplier replica contents to the consumer server.

   See Section 11.15.4.2, "Importing the LDIF File to the Consumer Server" .

11.15.4.1. Exporting a Replica to LDIF

There are three ways to convert a replica database to LDIF:

- When creating a replication agreement, by selecting **Create consumer initialization file** in the **Initialize Consumer** dialog box of the **Replication Agreement Wizard**.
- From the Directory Server Console, by right-clicking the replication agreement under the **Replication** folder and choosing **Create LDIF File** from the pop-up menu.
- From the command line by using the export command, as described in Section 4.2.3, "Exporting to LDIF from the Command Line". Exporting to LDIF with any of the command-line tools requires using an option to export the database as a replica; this means that the exported LDIF contains the proper entries to initialize the consumer when the LDIF is imported.

  For the **db2ldif** and **db2ldif.pl** scripts, this is the **-r** option. For example:

  ```
  [root@server ~]# db2ldif -r -n database1 -a /export/output.ldif
  ```

  For the **cn=export,cn=tasks,cn=config** entry, this is the ***nsExportReplica*** attribute.

  ```
  ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

  dn: cn=example export,cn=export,cn=tasks,cn=config
  changetype: add
  objectclass: extensibleObject
  cn: example export
  nsInstance: userRoot
  nsFilename: /home/files/example.ldif
  nsExportReplica: true
  ```

11.15.4.2. Importing the LDIF File to the Consumer Server

Import the LDIF file which contains the supplier replica contents to the consumer server by using the import features in the Directory Server Console or by using either the **ldif2db** script or **ldif2db.pl** script. Both import methods are described in Section 4.1.6, "Importing from the Command Line".

NOTE

With the **ldif2db.pl** script, the LDIF file import operation does not require a server restart. For more information on command-line scripts, see the *Directory Server Configuration and Command-Line Tool Reference*.

11.15.5. Filesystem Replica Initialization

A very large database, such as one with several million entries, can take an hour or more to initialize a consumer from the Console or even with manual initialization. To save time, use *filesystem replica initialization*.

Directory Server has the capability to initialize a replica using the database files from the supplier server. This avoids the need to rebuild the consumer database and can be done at essentially the speed of the network between the two servers by transferring the files with FTP or NFS, for example. Instead of sending entries using LDAP to replica servers, filesystem replica initialization populates the new database on the destination server by *backing up* the supplier database on one server and *restoring* the database on the destination server.

This method of initializing consumers is especially useful in replication over wide-area networks or over networks with slow or unstable connections.

For smaller databases, Red Hat recommends using manual initialization or initialize consumers from the Console.

NOTE

The destination server must have the same architecture and the same bit size as the supplier server for the initialization to succeed. For example, Red Hat Enterprise Linux 6 (64-bit) to Red Hat Enterprise Linux 6 (64-bit).

11.16. FORCING REPLICATION UPDATES

When a Directory Server involved in replication is stopped for regular maintenance, it must be updated immediately when it comes back online. In the case of a supplier in a multi-master environment, the directory information needs to

be updated by the other supplier in the multi-master set. In other cases, when a hub or a dedicated consumer is taken offline for maintenance, when they come back online, they need to be updated by the supplier server.

Even if the replication agreements are configured to keep the supplier and consumer servers always in sync, it is not sufficient to bring back up-to-date a server that has been offline for over five minutes. The **Always Keep in Sync** option means that the server generates a replication operation for every update operation it processes. However, if this replication operation cannot be performed because the consumer is offline, the operation times out after 10 minutes.

NOTE

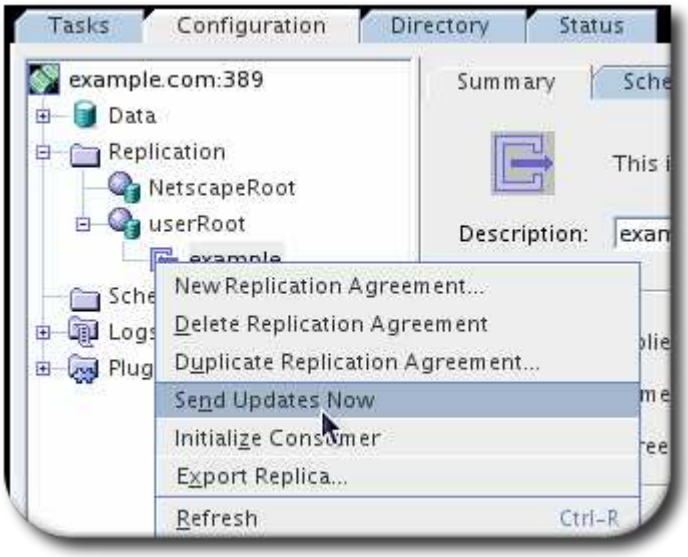
The procedures described in this section can only be used when replication is already set up and consumers have been initialized.

To ensure that directory information will be synchronized immediately when a server comes back online, use either the Directory Server Console on the supplier server that holds the reference copy of the directory information or a customizable script.

11.16.1. Forcing Replication Updates from the Console

To ensure that replication updates are sent immediately when a consumer or a supplier in a multi-master replication configuration comes back online after a period of time, do the following on the supplier server that holds the most recent version of the directory information:

1. In the Directory Server Console, click the **Configuration** tab, expand the **Replication** folder and database nodes, and select the replication agreement corresponding to the replica to update.
2. Right click the replication agreement, and choose **Send Updates Now** from the drop-down list.

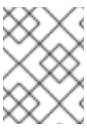


This initiates replication toward the server that holds the information that needs to be updated.

11.16.2. Forcing Replication Updates from the Command Line

From the consumer that requires updating, run a script that prompts the supplier to send replication updates immediately. This script is shown in Example 11.5, "replicate_now.sh Script Example".

Copy this example script and name it something like **replicate_now.sh**. Substitute the actual values for the variables listed in Example 11.5, "replicate_now.sh Script Example".



NOTE

This script must be run manually since it cannot be configured to run automatically as soon as the server, which was offline, comes back online again.

Example 11.5. replicate_now.sh Script Example

```
#!/bin/sh
```

```
SUP_HOST=supplier_hostname
SUP_PORT=supplier_portnumber
SUP_MGRDN=supplier_directoryManager
SUP_MGRPW=supplier_directoryManager_password
MY_HOST=consumer_hostname
MY_PORT=consumer_portnumber

ldapsearch -x -LLL -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \
   -w ${SUP_MGRPW} -b "cn=mapping tree,cn=config" \
   "(&(objectclass=nsds5replicationagreement) \
      (nsDS5ReplicaHost=${MY_HOST}) \
      (nsDS5ReplicaPort=${MY_PORT}))" \
   -o ldif-wrap=no dn | grep "^dn: " > /tmp/$$.dn

cp /tmp/$$.dn /tmp/$$.off
cp /tmp/$$.dn /tmp/$$.on

cat >> /tmp/$$.off <<EOF
changetype: modify
replace: nsds5ReplicaEnabled
nsds5ReplicaEnabled: off
EOF

cat >> /tmp/$$.on<<EOF
changetype: modify
replace: nsds5ReplicaEnabled
nsds5ReplicaEnabled: on
EOF

ldapmodify -x -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \
   -w ${SUP_MGRPW} -f /tmp/$$.off

sleep 1

ldapmodify -x -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \
   -w ${SUP_MGRPW} -f /tmp/$$.on

rm -f /tmp/$$.*
```

Table 11.4. Replicate_Now Variables

Variable	Definition
supplier_hostname	Hostname of the supplier to contact for information on replication agreements with the current consumer.
supplier_portnumber	LDAP port in use on the supplier.
supplier_directoryManager	DN of the privileged Directory Manager user on the supplier.
supplier_directoryManager_password	Password of the privileged Directory Manager user on the supplier.
consumer_hostname	Hostname of the current consumer.
consumer_portnumber	LDAP port in use on the consumer.

For the update operation to occur over an SSL connection, modify the **ldapmodify** command in the script with the appropriate parameters and values. For more information on the **ldapmodify** command, see Section 3.2, “Managing Entries from the Command Line”.

11.17. REPLICATION OVER SSL

The Directory Servers involved in replication can be configured so that all replication operations occur over an SSL connection. To use replication over SSL:

- Configure both the supplier and consumer servers to use SSL.

- Configure the consumer server to recognize the supplier server's certificate as the supplier DN. Do this only to use SSL client authentication rather than simple authentication.

These procedures are described in Section 7.4, "Setting up TLS/SSL".

If attribute encryption is enabled, a secure connection is required for replication.

NOTE

Replication configured over SSL with certificate-based authentication will fail if the supplier's certificate is only capable of behaving as a server certificate, and not also a client during an SSL handshake. Replication with certificate-based authentication uses the Directory Server's server certificate for authentication to the remote server.

When the servers are configured to use SSL, configure an SSL connection for replication in the **Replication Agreement Wizard**. The **Source and Destination** sets how to bind between the supplier and the consumer, and this is where SSL is set.

There are two ways to use SSL for replication:

- Select **SSL Client Authentication**.

  With SSL client authentication, the supplier and consumer servers use certificates to authenticate to each other.

- Select **Simple Authentication**.

  With simple authentication, the supplier and consumer servers use a bind DN and password to authenticate to each other, which are supplied in the **Replication Agreement Wizard** text fields provided. Simple authentication takes place over a secure channel but without certificates.

NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

Once a replication agreement is created, the connection type (SSL or non SSL) cannot be changed in the agreement because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

Also, the port listed for the consumer is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.

11.18. SETTING REPLICATION TIMEOUT PERIODS

Suppliers must have an exclusive connection to a consumer to send updates to the directory. As mentioned in Section 11.5.4, "Preventing Monopolization of the Consumer in Multi-Master Replication" , it is possible to configure a wait time for suppliers attempting to connect to a consumer, so that the supplier does not hang while the consumer is tied up with another supplier.

It is also possible to set a timeout period for a supplier, so that it does not stay connected to a consumer interminably attempting to send updates over a slow or broken connection.

There are two attributes which set the timeout period:

- ***nsDS5ReplicaTimeout*** sets the number of seconds that the replication operation waits for a response from the consumer before timing out and failing. To set the optimum number, check the access logs to see the average amount of time that the replication process takes, and set the timeout period accordingly.

- ***nsDS5DebugReplicaTimeout*** sets the timeout period for the replication operation when debug logging is enabled. This setting may be appreciably higher than the ***nsDS5ReplicaTimeout*** setting because debug logging can slow down directory operations. This attribute can optionally set an error log level where this parameter is applied; the default is replication debugging (8192).

NOTE

The timeout period is limited to the maximum 32-bit integer in seconds, which translates to 24.8 days.

Both of these attributes are set in the configuration for the replicated suffix. For example, this sets timeout periods for the **ou=People** suffix:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=replica,cn=ou=People\,dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
add: nsDS5ReplicaTimeout
nsDS5ReplicaTimeout: 600
add: nsDS5DebugReplicaTimeout
nsDS5DebugReplicaTimeout: 6000
```

11.19. REPLICATING O=NETSCAPEROOT FOR ADMIN SERVER FAILOVER

Replication usually occurs between Directory Server user databases to distribute directory data, but it is also possible to use replication to provide failover support for the Admin Server database, **o=NetscapeRoot**.

1. Install and configure the first Directory Server instance.

   The **setup-ds-admin.pl** script has an option, **-f**, which references an **inf**. The **inf** can be used to import LDIF files through the **ConfigFile** parameter, and the LDIF files can create databases, suffixes, and replication entries. (The **inf** file is described in more detail in the *Directory Server Installation Guide*.)

   ```
   [root@server ~]# setup-ds-admin.pl -f /tmp/server1.inf
   ```

   To configure the **o=NetscapeRoot** database on **server1** as a multi-master supplier replica, use the following statements in the **inf** file:

   ```
   [slapd]
   ...
   ConfigFile = repluser.ldif Example 11.1, “Example Supplier Bind DN Entry”
   ConfigFile = changelog.ldif Example 11.2, “Example Changelog Entry”
   ConfigFile = replica.ldif Example 11.3, “Example Supplier Replica Entry”
   ConfigFile = replagreement.ldif Example 11.4, “Example Replication Agreement Entry”
   ...
   ```

2. Install and configure the second Directory Server instance. For the second server, **server2.example.com**, use the **setup-ds.pl** command, which installs a Directory Server instance without installing a local Admin Server.

   ```
   [root@server ~]# setup-ds.pl -f /tmp/server2.inf
   ```

   With server2, use the **inf** file to create and configure a **o=NetscapeRoot** database on **server2** as a multi-master supplier replica:

   ```
   [slapd]
   ...
   ConfigFile = netscaperootdb.ldif Example 2.1, “Example Root Suffix Entry”
   ConfigFile = repluser.ldif Example 11.1, “Example Supplier Bind DN Entry”
   ConfigFile = changelog.ldif Example 11.2, “Example Changelog Entry”
   ConfigFile = replica.ldif Example 11.3, “Example Supplier Replica Entry”
   ConfigFile = replagreement.ldif Example 11.4, “Example Replication Agreement Entry”
   ...
   ```

3. Initialize the **o=NetscapeRoot** database on **server2** from **server1**. Add the ***nsds5replicarefresh*** attribute to the replication agreement on **server1**.

   ```
   [root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h supplier1.example.com

   dn: cn=ExampleAgreement1,cn=replica,cn=o=NetscapeRoot,cn=mapping tree,cn=config
   changetype: modify
   replace: nsds5beginreplicarefresh
   nsds5beginreplicarefresh: start
   ```

4. Run the **register-ds-admin.pl** to create a local Admin Server on **server2** and switch the configuration directory for **server2** to its own **o=NetscapeRoot** database from **server1**.

```
[root@server ~]# register-ds-admin.pl
```

5. Add the following access control instructions (ACI) on **server2**, to enable members of the **Configuration Administrators Group**, the server instance entry **SIE group**, and the **admin** user, to run on suffixes belonging to **server2**. For example, to run on the **dc=example,dc=com** suffix, enter:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h server2.example.com
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "Configuration Administrators Group";
 allow (all) groupdn="ldap:///cn=Configuration Administrators,ou=Groups,
 ou=TopologyManagement,o=NetscapeRoot";)
-
add: aci
aci: (targetattr="*")(version 3.0; acl "Configuration Administrator";
 allow (all) userdn="ldap:///uid=admin,
 ou=Administrators,ou=TopologyManagement,o=NetscapeRoot";)
-
add: aci
aci: (targetattr = "*")(version 3.0; acl "SIE Group";  allow (all) groupdn =
 "ldap:///cn=slapd-instance,cn=Red Hat Directory Server,cn=Server Group,
 cn=machine_name,ou=example.com,o=NetscapeRoot";)
```

6. Disable the PTA Plug-in on **server2** so that it does not pass bind operations for the administrative users in its **o=NetscapeRoot** to **server1**.

   See Section 1.8.1, "Enabling Plug-ins in the Directory Server Console" .

11.20. REPLICATION WITH EARLIER RELEASES

This section provides information on how to optimize replication with earlier releases of Directory Server.

11.20.1. Using Legacy Replication

Directory Server 9.0 can be involved in replication with earlier releases of Directory Server, providing the following conditions are met:

- Directory Server 9.0 is a consumer.
- The legacy suppliers can be Directory Server 4.0, 4.1, and 4.1x.

The following restrictions apply:

- A legacy Directory Server and Directory Server 9.0 cannot update the same replica. However, this version of Directory Server can have different replicas, where one is supplied by a legacy Directory Server and the other is supplied by Directory Server 9.0.
- Directory Server 9.0 cannot be a supplier for other replicas.

The main advantage of using Directory Server 9.0 as a consumer of a legacy Directory Server is to ease the migration of a replicated environment, especially since migration is not supported from 4.x servers to 9.0. For more information on migration, see the *Directory Server Installation Guide*.

11.20.2. Legacy Replication and Parent Object Classes

There is one important difference between the way that Directory Server 4.x servers handle replicated entries and the way that Directory Server 9.0 handles replicated entries. In Directory Server 4.x, entries could be added without specifying parent object classes, and when those entries were modified or replicated, the server would not automatically insert those parent object classes. For example, a user could be added with the **inetorgperson** object class, but not the **top** or **person** object classes:

```
dn: uid=jsmith,ou=People,dc=example,dc=com
objectclass: inetorgperson
uid: jsmith
```

```
cn: John Smith
sn: Smith
```

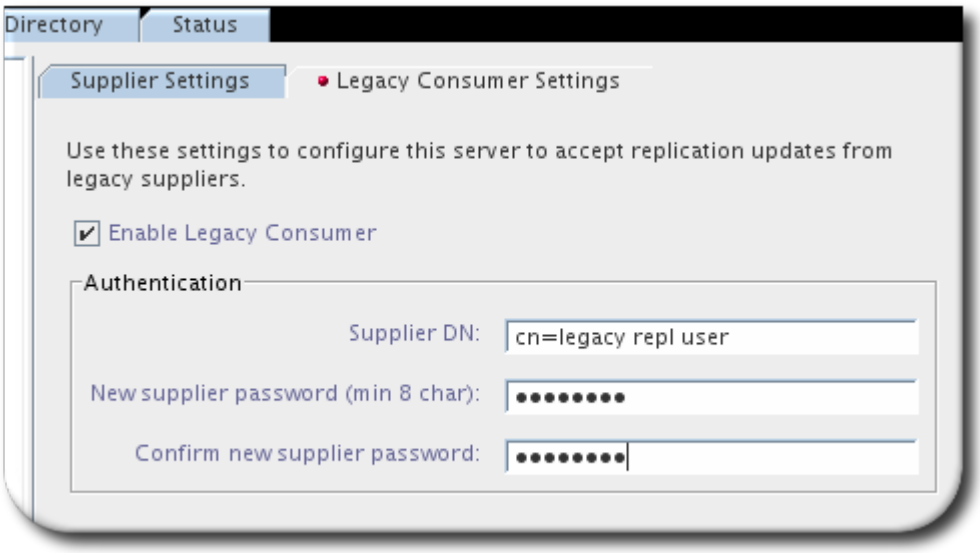
However, in Directory Server 9.0, the parent object classes are automatically added to the entry when the entry is added, modified, or replicated. This means that the entries will be slightly different on the Directory Server 4.x supplier and the Directory Server 9.0 consumer, because the Directory Server 9.0 entry will have the parent object classes:

```
dn: uid=jsmith,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: inetorgperson
uid: jsmith
cn: John Smith
sn: Smith
```

11.20.3. Configuring Legacy Replication

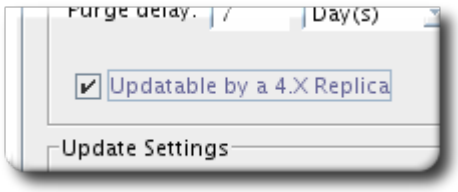
To set up legacy replication:

1. In the Directory Server Console, click the **Configuration** tab.
2. Select the **Replication** node, and click the **Legacy Consumer Settings** tab in the right pane.
3. Check the **Enable Legacy Consumer** check box.



   This activates the fields in the **Authentication** box.
4. Specify the supplier bind DN that the legacy supplier server will use to bind to the consumer.

   Optionally, specify a password at least 8 characters long.
5. Click **Save**.
6. Now configure legacy consumer settings for each replica that will receive updates from a legacy supplier.
   1. In the navigation tree, expand the **Replication** node, and select a replica that will receive updates from the legacy supplier.
   2. In the **Common Settings** area, select the **Enable Replica** and **Updatable by a 4.x Replica** check boxes.



These options are the only ones required for replication to work. Optionally, specify a replica ID. It is not necessary to specify a supplier DN because the one specified in step 4 will be used.

3. Click **Save**.

7. Repeat step 6 for each read-only replica that will receive updates from a legacy supplier.

8. To complete the legacy replication setup, configure the legacy supplier to replicate to the Directory Server 9.0 instance. For instructions on configuring a replication agreement on a 4.x Directory Server, see the documentation for the legacy Directory Server.

NOTE

The Directory Server Console will not prevent you from configuring a database as a read-write replica and enabling legacy consumer settings. This makes migration easier because the Directory Server can be configured as it should be after the migration and legacy consumer settings only have to be active for the duration of the transition.

11.21. USING THE RETRO CHANGELOG PLUG-IN

The Retro Changelog plug-in configures Directory Server to maintain a changelog that is compatible with the changelog implemented in Directory Server 4.0, 4.1, and 4.1x. Maintaining a retro changelog is essential to maintain a changelog for directory clients that depend on a Directory Server 4.x-style changelog.

To use the retro changelog plug-in, the Directory Server 9.0 instance must be configured as a single-master replica.

When the Directory Server is configured to maintain a retro changelog, this changelog is stored in a separate database under a special suffix, **cn=changelog**.

The retro changelog consists of a single level of entries. Each entry in the changelog has the object class ***changeLogEntry*** and can include the attributes listed in Table 11.5, “Attributes of a Retro Changelog Entry” .

Table 11.5. Attributes of a Retro Changelog Entry

Attribute	Definition
changeNumber	This single-valued attribute is always present. It contains an integer which uniquely identifies each change. This number is related to the order in which the change occurred. The higher the number, the later the change.
targetDN	This attribute contains the DN of the entry that was affected by the LDAP operation. In the case of a **modrdn** operation, the ***targetDN*** attribute contains the DN of the entry before it was modified or moved.
changetype	Specifies the type of LDAP operation. This attribute can have a value of add, delete, modify, or **modrdn**.
changes	For add and modify operations, contains the changes made to the entry in LDIF format.
newrdn	In the case of **modrdn** operations, specifies the new RDN of the entry.
deleteoldrdn	In the case of **modrdn** operations, specifies whether the old RDN was deleted.
newSuperior	In the case of **modrdn** operations, specifies the ***newSuperior*** attribute of the entry.

This section contains information on the following retro changelog items:

- Section 11.21.1, “Enabling the Retro Changelog Plug-in”
- Section 11.21.2, “Trimming the Retro Changelog”

- Section 11.21.3, “Searching and Modifying the Retro Changelog”
- Section 11.21.4, “Retro Changelog and the Access Control Policy”

11.21.1. Enabling the Retro Changelog Plug-in

The retro changelog plug-in configuration information is stored in the **cn=Retro Changelog Plugin,cn=plugins,cn=config** entry in **dse.ldif**. To enable the retro changelog plug-in from the command line:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Use the **ldapmodify** command to import the LDIF file into the directory.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x -f retro.ldif
```

3. Restart the server.

   For information on restarting the server, see Section 1.3, “Starting and Stopping Servers”.

The retro changelog is created in the directory tree under a special suffix, **cn=changelog**.

The procedure for enabling the retro changelog plug-in from Directory Server Console is the same as for all Directory Server plug-ins. For information, see Section 1.8.1, “Enabling Plug-ins in the Directory Server Console” .

11.21.2. Trimming the Retro Changelog

The size of the retro changelog is automatically reduced if you lower the maximum age of records set in the ***nsslapd-changelogmaxage*** parameter and the next trim interval, set in ***nsslapd-changelog-trim-interval***, is executed.

For example, to set maximum age of records in the retro changelog to two days:

```
# ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 2d
```

11.21.3. Searching and Modifying the Retro Changelog

The changelog supports search operations and is optimized for searches that include filters of the form **(&(changeNumber>=X)(changeNumber<=Y))**.

As a general rule, do not perform add or modify operations on the retro changelog entries, although entries can be deleted to trim the size of the changelog. Only modify the retro changelog entry to modify the default access control policy.

11.21.4. Retro Changelog and the Access Control Policy

When the retro changelog is created, the following access control rules apply by default:

- Read, search, and compare rights are granted to all authenticated users (**userdn=anyone**, not to be confused with anonymous access where **userdn=all**) to the retro changelog top entry **cn=changelog**.
- Write and delete access are not granted, except implicitly to the Directory Manager.

Do not grant read access to anonymous users because the changelog entries can contain modifications to sensitive information, such as passwords. Only authenticated applications and users should be allowed to access this information.

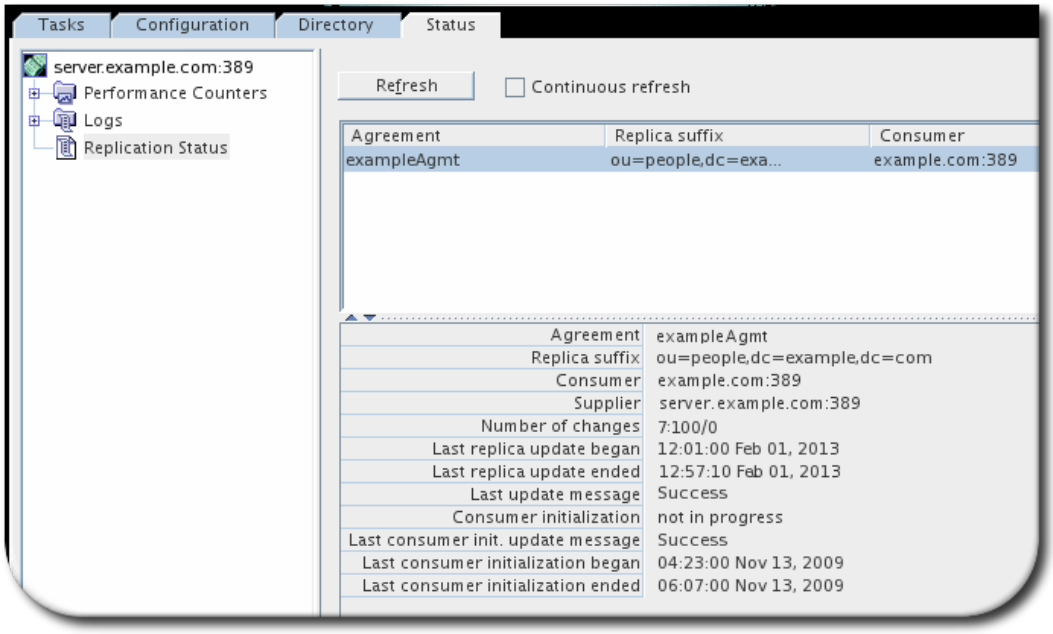
To modify the default access control policy which applies to the retro changelog, modify the ***aci*** attribute of the **cn=changelog** entry.

11.22. MONITORING REPLICATION STATUS

The replication status can be viewed in the Directory Server Console or Red Hat Administration Express (Section 11.22.2, "Monitoring Replication from Admin Express").

11.22.1. Monitoring Replication Status from the Console

1. Select the **Status** tab, and then, in the left navigation tree, select **Replication Status**.



   In the right pane, a table appears that contains information about each of the replication agreements configured for this server.

2. Click **Refresh** to update the contents of the tab.

   The status information displayed is described in Table 11.6, "Directory Server Console Replication Status".

Table 11.6. Directory Server Console Replication Status

Table Header	Description
Agreement	The name of the replication agreement.
Replica suffix	The suffix that is replicated.
Supplier	The supplier server in the agreement.
Consumer	The consumer server in the agreement.
Number of changes	A ratio showing the changes sent to this replica since the server started. This value has the format *replica_id:changes_sent/changes_skipped*. So, if the replica ID is 7, 100 changes were sent, and no changes were skipped, the value of the number of changes is **7:100/0**.
Last replica update began	The time when the most recent replication update started.
Last replica update ended	The time when the most recent replication update ended.
Last update message	The status for the most recent replication updates.
Consumer initialization	The current status on consumer initialization (in progress or not).

Table Header	Description
Last consumer initialization update message	The status on the last initialization of the consumer.
Last consumer initialization began	The time when the initialization of the consumer replica started.
Last consumer initialization ended	The time when the initialization of the consumer replica ended.

11.22.2. Monitoring Replication from Admin Express

Admin Express has an option to monitor replication status in real-time, meaning that it shows the number of updates, times the most recent updates were sent, error and success messages, replication schedule, the replicated directory suffix, and other information. Unlike other ways of checking replication status, the Admin Express **Replication Status** page shows the real-time status of replication, including updates in progress, current changes sequence numbers, and the lag between when a change is made on the supplier and when that change is sent to the consumer.

Monitoring replication is set up using a simple configuration file which specifies which server to monitor and what supplier and consumer replicas to include in the status page.

When trying to monitor replication status through Admin Express, remember two things:

- The **Replication Status** page is only available for supplier servers. (It can be opened for other types of replicas; there is just no information available and has the message *The server is not a master or it has no replication agreement*.)
- The configuration file must be in a directory that is accessible to Admin Server, and the file must be readable by the Admin Server user. By default, the user is **nobody**. If you set a different account during the installation, like Red Hat recommends, use this account instead for a better security.

  The user is set in the **console.conf** file. To check the user, use **grep** to return the value:

  ```
  grep \^User /etc/dirsrv/admin-serv/console.conf
  ```

  The configuration file should be readable by the Admin Server user and no other users, so consider resetting the permissions on the file:

  ```
  chmod 0400 filename
  ```

To view in-progress status of replication in Admin Express:

1. Create a configuration file. The configuration file lists all of the servers to monitor for replication, giving their host name or IPv4 or IPv6 address, port, the bind credentials to use, and then optional settings for aliases and time lag colors.

   ```
   #Configuration File for Monitoring Replication Via Admin Express
    [connection] Required. Gives the server host (or IPv4 or IPv6 address), port, supplier bind DN, and password.
   host1.example.com:389:cn=replication manager:mypassword
   host2.example.com:3891:cn=replication manager:altpassword

   [alias] Optional. Gives a friendly-name alias to the servers and consumers.
   M1 = host1.example.com:389
   M2 = host2.example.com:3891
   C1 = host3.example.com:3892
   C2 = host4.example.com:3890

   [color] Optional. Sets the color for the time lag boxes.
   0 = #ccffcc
   5 = #FFFFCC
   60 = #FFCCCC
   ```

   The configuration file must be in a directory that is accessible to the Admin Server, and the file must be readable by the Admin Server user. By default, the user is **nobody**.

   The user is set in the **console.conf** file. To check the user, use **grep** to return the value:

```
grep \^User /etc/dirsrv/admin-serv/console.conf
```

The configuration file should be readable by the Admin Server user and no other users, so consider resetting the permissions on the file:

```
chmod 0400 filename
```

2. In the Admin Server web page, click the **Admin Express** link, and log in.
3. Click the **Replication Status** link by the supplier server name.
4. Type the path to the configuration file in the **Configuration file** field. Also, set the refresh rate, which is how frequently the replication status page updates; the default is 300 seconds.

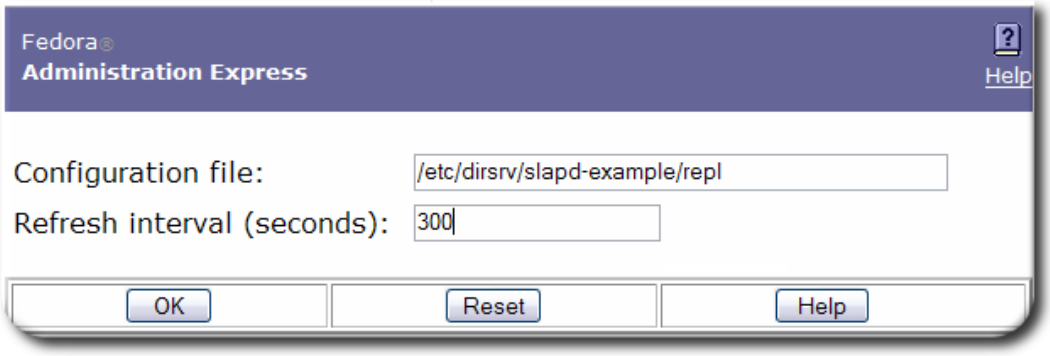


Figure 11.5. Viewing Replication Status

5. Click **OK**.

The **Replication Status** page shows the status for sending updates to every consumer listed in the configuration file.

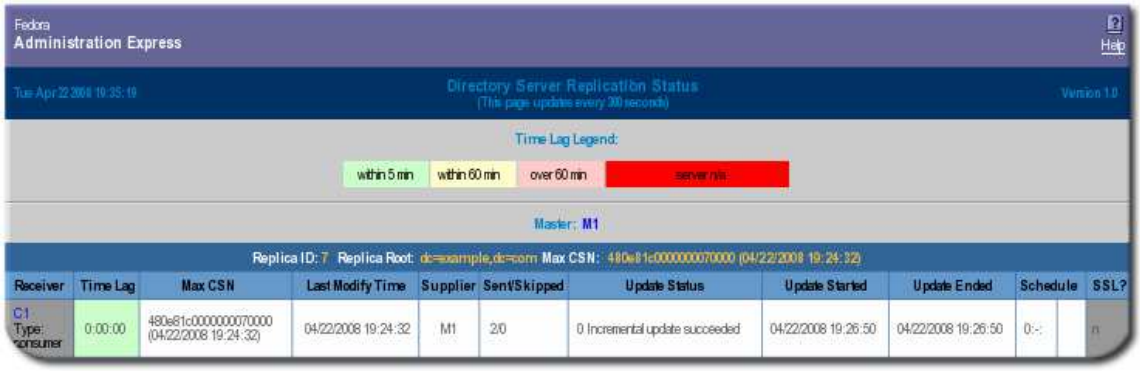


Figure 11.6. Viewing Replication Status

Table	Description
Table header	The table header shows the replica ID of the supplier replica, the replicated suffix root (such as **dc=example,dc=com**), and the maximum change state number (CSN) on the supplier. (The CSN is the ID of the latest change on the supplier, while the max CSN for the supplier shows the last update it received.)
Max CSN	The ID number of the most recent CSN the consumer has received that originated from the supplier.
Time lag	How long it takes for the consumer to receive updates from the supplier; this is the time difference between the supplier and the consumer's max CSNs. When a consumer is in sync with its supplier, the time lag is **0**.

Table	Description
Last Modify Time	Gives the time of the last update for the consumer (the time the last CSN entry was sent).
Supplier	Gives the name of the supplier sending updates to that consumer; this can be useful if a consumer receives updates from multiple suppliers or there are multiple suppliers being monitored on the **Replication Status** page.
Sent/Skipped	The number of changes that were sent from the supplier and the number skipped in the replication update. The numbers are kept in suppliers' memory only and are cleared if the supplier is restarted.
Update Status	The status code (and meaning) for the last update. This column can indicate a possible deadlock if *all* the suppliers complain that they cannot acquire a busy replica. It is normal for there to be a busy message if one of the suppliers is doing an update.
Update Start and End	The timestamps for when the most recent update process started and ended.
Schedule	The configured replication schedule. **0:-:** means that the consumer is continually updated by the supplier.
SSL?	Indicates whether the supplier connects to the consumer over SSL.

11.23. SETTING REPLICATION SESSION HOOKS

Client applications can have some control over when the Directory Server performs replication by using custom plug-ins that define *replication session hooks*. The hooks apply to both the master sending the information and the consumer receiving the information. Basically, the master sends a preliminary message to the consumer and the consumer sends a response message. If either side sends a message that does not meet the expected criteria (such as the server version or some schema requirement), then the replication operation is terminated.

The intent of replication session hooks is to require some kind of parity between the servers involved in replication. If there is a situation that could cause conflicts or data corruption – such as different server versions which use different sets of schema – then the session hooks identify that potential problem. This prevents replication when it could hurt server performance.

The session hooks are implemented through callback functions in the Directory Server replication functions.

The call backs are defined in an ordered array that follows the replication operation process from initializing the replication agreement to receiving the response from the consumer.

```
static void *test_repl_session_api[] = {
  NULL, /* reserved for api broker use, must be zero */
  test_repl_session_plugin_agmt_init_cb,
  test_repl_session_plugin_pre_acquire_cb,
  test_repl_session_plugin_reply_acquire_cb,
  test_repl_session_plugin_post_acquire_cb,
  test_repl_session_plugin_recv_acquire_cb,
  test_repl_session_plugin_destroy_cb
};
```

Then, the custom replication plug-in registers the callbacks. In this example, it requires a certain version of the server.

```
int test_repl_session_plugin_init(Slapi_PBlock *pb)
{
...
  if( slapi_apib_register(REPL_SESSION_v1_0_GUID, test_repl_session_api) ) {
      slapi_log_error( SLAPI_LOG_FATAL, test_repl_session_plugin_name,
                  "<-- test_repl_session_plugin_start -- failed to register repl_session api -- end\n");
      return -1;
```

```
    }
...
}
```

NOTE

The callbacks themselves are described in the *Plug-in Programmer's Guide*. This example is provided to make administrators and programmers aware that the replication process can be controlled by using these session hooks.

11.24. SOLVING COMMON REPLICATION CONFLICTS

Multi-master replication uses a loose consistency replication model. This means that the same entries can be changed on different servers. When replication occurs between the two servers, the conflicting changes need to be resolved. Mostly, resolution occurs automatically, based on the timestamp associated with the change on each server. The most recent change takes precedence.

However, there are some cases where change conflicts require manual intervention in order to reach a resolution. Entries that have a change conflict that cannot be resolved automatically by the replication process contain a conflict marker attribute ***nsds5ReplConflict***. The ***nsds5ReplConflict*** attribute is an operational attribute which is indexed for presence and equality, so it is simple to search for entries that contain this attribute. For example:

```
ldapsearch -D adminDN -W
    -b "dc=example,dc=com" "nsds5ReplConflict=*" \* nsds5ReplConflict
```

The ***nsds5ReplConflict*** attribute is already indexed for presence and equality, but for performance reasons, if there are many conflicting entries every day, index the ***nsds5ReplConflict*** attribute in other indexes. For information on indexing, see Chapter 9, *Managing Indexes*.

- Section 11.24.1, "Solving Naming Conflicts"
- Section 11.24.2, "Solving Orphan Entry Conflicts"
- Section 11.24.3, "Solving Potential Interoperability Problems"

11.24.1. Solving Naming Conflicts

When two entries are created with the same DN on different servers, the automatic conflict resolution procedure during replication renames the last entry created, including the entry's unique identifier in the DN. Every directory entry includes a unique identifier given by the operational attribute ***nsuniqueid***. When a naming conflict occurs, this unique ID is appended to the non-unique DN.

For example, the entry **uid=adamss,ou=people,dc=example,dc=com** is created on Server A at time **t1** and on Server B at time **t2**, where **t2** is greater (or later) than **t1**. After replication, Server A and Server B both hold the following entries:

- **uid=adamss,ou=people,dc=example,dc=com** (created at time **t1**)
- **nsuniqueid=66446001-1dd211b2+uid=adamss,dc=example,dc=com** (created at time **t2**)

The second entry needs to be renamed in such a way that it has a unique DN. The renaming procedure depends on whether the naming attribute is single-valued or multi-valued.

11.24.1.1. Renaming an Entry with a Multi-Valued Naming Attribute

To rename an entry that has a multi-valued naming attribute:

1. Rename the entry using a new value for the naming attribute, and keep the old RDN. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: nsuniqueid=66446001-1dd211b2+uid=adamss,dc=example,dc=com
 changetype: modrdn
 newrdn: uid=NewValue
 deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: uid=NewValue,dc=example,dc=com
```

```
changetype: modify
delete: uid
uid: adamss
-
delete: nsds5ReplConflict
-
```

NOTE

The unique identifier attribute ***nsuniqueid*** cannot be deleted.

The Console does not support editing multi-valued RDNs. For example, if there are two servers in a multi-master mode, an entry can be created on each server with the same user ID, and then the new entries' RDN changed to the ***nsuniqueid uid*** value. Attempting to modify this entry from the Console returns the error *Changes cannot be saved for entries with multi-valued RDNs.*

Opening the entry in the advanced mode shows that the naming attribute has been set to ***nsuniqueid uid***. However, the entry cannot be changed or corrected by changing the user ID and RDN values to something different. For example, if **jdoe** was the user ID and it should be changed to **jdoe1**, it cannot be done from the Console. Instead, use the **ldapmodify** command:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=John Doe
changetype: modify
replace: uid
uid: jdoe

dn: cn=John Doe
changetype: modrdn
newrdn: uid=jdoe1
deleteoldrdn: 1
```

11.24.1.2. Renaming an Entry with a Single-Valued Naming Attribute

To rename an entry that has a single-valued naming attribute:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: nsuniqueid=66446001-1dd211b2+dc=pubs,dc=example,dc=com
 changetype: modrdn
 newrdn: cn=TempValue
 deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: cn=TempValue,dc=example,dc=com
 changetype: modify
 delete: dc
 dc: pubs
 -
 delete: nsds5ReplConflict
 -
```

NOTE

The unique identifier attribute ***nsuniqueid*** cannot be deleted.

3. Rename the entry with the intended attribute-value pair. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
 dn: cn=TempValue,dc=example,dc=com
 changetype: modrdn
 newrdn: dc=NewValue
 deleteoldrdn: 1
```

Setting the value of the ***deleteoldrdn*** attribute to **1** deletes the temporary attribute-value pair **cn=***TempValue*. To keep this attribute, set the value of the ***deleteoldrdn*** attribute to **0**.

11.24.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a **glue** entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

Glue entries are temporary entries that include the object classes **glue** and **extensibleObject**. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the **glue** object class and the ***nsds5ReplConflict*** attribute.

  In such cases, either modify the glue entry to remove the **glue** object class and the ***nsds5ReplConflict*** attribute to keep the entry as a normal entry or delete the glue entry and its child entries.

- The server creates a minimalistic entry with the **glue** and **extensibleObject** object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

11.24.3. Solving Potential Interoperability Problems

For reasons of interoperability with applications that rely on attribute uniqueness, such as a mail server, it may be necessary to restrict access to the entries which contain the ***nsds5ReplConflict*** attribute. If access is not restricted to these entries, then the applications requiring one attribute only pick up both the original entry and the conflict resolution entry containing the ***nsds5ReplConflict***, and operations will fail.

To restrict access, modify the default ACI that grants anonymous read access:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: dc=example,dc=com
changetype: modify
delete: aci
aci: (target ="ldap:///dc=example,dc=com")(targetattr
   !="userPassword")(version 3.0;acl "Anonymous read-search
   access";allow (read, search, compare)(userdn = "ldap:///anyone");)
-
add: aci
aci: (target="ldap:///dc=example,dc=com")(targetattr!="userPassword")
   (targetfilter="(!(nsds5ReplConflict=*))")(version 3.0;acl
   "Anonymous read-search access";allow (read, search, compare)
   (userdn="ldap:///anyone");)
-
```

The new ACI filters out all entries that contain the ***nsds5ReplConflict*** attribute from search results.

11.24.4. Resolving Errors for Obsolete/Missing Suppliers

Information about the replication topology – all of the suppliers which are supplying updates to each other and other replicas within the same replication group – is contained in a set of metadata called the *replica update vector (RUV)*. The RUV contains information about the supplier like its ID and URL, its latest change state number for changes made on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

When one supplier is removed from the replication topology, it may remain in another replica's RUV. When the other replica is restarted, it can record errors in its log that the replication plug-in does not recognize the (removed) supplier.

```
[09/Sep/2017:09:03:43 -0600] NSMMReplicationPlugin - ruv_compare_ruv: RUV [changelog max RUV] does not
 contain element [{replica 55 ldap://server.example.com:389} 4e6a27ca000000370000 4e6a27e8000000370000]
 which is present in RUV [database RUV]
......
[09/Sep/2017:09:03:43 -0600] NSMMReplicationPlugin - replica_check_for_data_reload: Warning: for replica
 dc=example,dc=com there were some differences between the changelog max RUV and the database RUV.  If
```

there are obsolete elements in the database RUV, you should remove them using the CLEANRUV task. If they are not obsolete, you should check their status to see why there are no changes from those servers in the changelog.

When the supplier is permanently removed from the topology, then any lingering metadata about that supplier should be purged from every other supplier's RUV entry.

There are three ways to do this:

- Removed from *all* suppliers in the topology using the **CLEAN*ALL*RUV** replication task.
- Removed from a single supplier in the topology (because of local errors) in using the **CLEANRUV** replication task.
- Removed from all suppliers in the topology using the directory task.

11.24.4.1. Removing an Obsolete Replica from a Single Supplier

If a server is offline or unavailable when a supplier is removed from the topology, it may not receive any updates that the other supplier was removed. In that case, the supplier's RUV would still contain entries about the missing supplier and would return missing element errors.

This can be cleaned up on a single supplier using the **CLEANRUV** replication task. The ***nsds5Task*** attribute identifies a replication-related task in a replica configuration entry. This attribute is generally added and removed automatically by the server as it performs regular replication tasks. However, the attribute can be added to a replica configuration entry to manually initiate a task, and it is used to clean obsolete supplier data from the local server's RUV.

To purge an obsolete supplier from the RUV:

1. Running the **CLEANRUV** replication task requires that old replica configuration DN and the old replica ID.
    1. Get the replica configuration entry DN by checking for replica entries in the **cn=mapping tree,cn=config** entry.

        ```
        ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b cn=config objectclass=nsds5replica

        dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
        cn: replica
        ...
        ```

    2. Get the replica ID. This is in the ***nsds5replicaid*** attribute in the configuration entry. The ID is also in the error message about the server being unable to find the replica, identified in the *element [{replica ID URL} uniqueId]* line. For example:

        ```
        [09/Sep/2011:09:03:43 -0600] NSMMReplicationPlugin - ruv_compare_ruv: RUV [changelog max
        RUV] does not
         contain element [{replica 55 ldap://server.example.com:389} 4e6a27ca000000370000
        4e6a27e8000000370000]
        ...
        ```

2. Use **ldapmodify** to replace the ***nsds5Task*** attribute in the configuration entry with **CLEANRUV** and the replica ID, in the form **CLEANRUV***#*. For example, for a replica with the ID of 55, the ***nsds5Task*** value is **CLEANRUV55**:

    ```
    ldapmodify -x -D "cn=directory manager" -W

    dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
    changetype: modify
    replace: nsds5task
    nsds5task: CLEANRUV55
    ```

11.24.4.2. Removing an Obsolete/Missing Supplier from All Servers in the Topology

If a supplier is taken offline without cleaning up its RUV entries, then all suppliers and hubs in the topology can register missing element errors in their replication logs.

This can be cleaned up on a single supplier using the **CLEANALLRUV** replication task. The ***nsds5Task*** attribute identifies a replication-related task in a replica configuration entry. This attribute is generally added and removed automatically by the server as it performs regular replication tasks. However, the attribute can be added to a replica

configuration entry to manually initiate a task, and it is used to clean obsolete supplier data from all RUV stores in the topology.

NOTE

The **CLEAN*ALL*RUV** task is replicated to all suppliers and hubs in the replication topology.

1. Running the **CLEANALLRUV** replication task requires that old replica configuration DN and the old replica ID.
    1. Get the replica configuration entry DN by checking for replica entries in the **cn=mapping tree,cn=config** entry.

```
ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b cn=config objectclass=nsds5replica

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
cn: replica
...
```

    2. Get the replica ID. This is in the ***nsds5replicaid*** attribute in the configuration entry.

```
ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b
cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config objectclass=nsds5replica
nsds5replicaid

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsds5replicaid: 55
...
```

    The ID is also in the error message about the server being unable to find the replica, identified in the *element [{replica ID URL} uniqueId]* line. For example:

```
[09/Sep/2011:09:03:43 -0600] NSMMReplicationPlugin - ruv_compare_ruv: RUV [changelog max
RUV] does not
 contain element [{replica 55 ldap://server.example.com:389} 4e6a27ca000000370000
4e6a27e8000000370000]
...
```

2. Use **ldapmodify** to add the ***nsds5Task*** attribute in the configuration entry with a value of **CLEANALLRUV** and the replica ID, in the form **CLEANALLRUV#**. For example, for a replica with the ID of 55, the ***nsds5Task*** value is **CLEANALLRUV55**:

```
ldapmodify -x -D "cn=directory manager" -W

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
replace: nsds5task
nsds5task: CLEANALLRUV55
```

11.24.4.3. Removing an Obsolete/Missing Supplier Using a Task Operation

If a supplier is taken offline without cleaning up its RUV entries, then all suppliers and hubs in the topology can register missing element errors in their replication logs.

There may be times when it is preferable to launch a directory task operation rather than a replication task. This can be done by creating an instance of the **cn=cleanallruv,cn=tasks,cn=config** task.

NOTE

As with the **CLEAN*ALL*RUV** replication task, this **cn=cleanruv,cn=tasks** operation is replicated to all suppliers and hubs in the replication topology.

1. Obtain the old replica configuration DN and the old replica ID.
    1. Get the replica configuration entry DN by checking for replica entries in the **cn=mapping tree,cn=config** entry.

```
ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b cn=config objectclass=nsds5replica

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
cn: replica
...
```

2. Get the replica ID. This is in the ***nsds5replicaid*** attribute in the configuration entry.

```
ldapsearch -xLLL -D "cn=directory manager" -W -s sub -b
cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config objectclass=nsds5replica
nsds5replicaid

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsds5replicaid: 55
...
```

2. Use **ldapmodify** to create the **cn=cleanallruv,cn=tasks,cn=config** entry. This task requires information on the replication configuration:
   - The base DN of the replicated database (***replica-base-dn***).
   - The replica ID (***replica-id***).
   - Whether to catch up to the max change state number (CSN) from the missing supplier or just remove all RUV entries and miss any updates (***replica-force-cleaning***); setting this to **no** means that the task catches up with all changes first, and then removes the RUV.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=clean 55,cn=cleanallruv,cn=tasks,cn=config
objectclass: extensibleObject
replica-base-dn: dc=example,dc=com
replica-id: 55
replica-force-cleaning: no
cn: clean 55
```

This task replicates to all servers in the topology. This task can also be aborted, since it can take several minutes to run, and the abort task is also propagated to all suppliers.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=abort 55,cn=abort cleanallruv,cn=tasks,cn=config
objectclass: extensibleObject
cn: abort 55
replica-base-dn: dc=example,dc=com
replica-id: 55
replica-certify-all: yes
```

The ***replica-certify-all*** attribute sets whether to wait for the task to be sent to all servers before completing on the local server.

11.25. TROUBLESHOOTING REPLICATION-RELATED PROBLEMS

This section lists some error messages, explains possible causes, and offers remedies.

It is possible to get more debugging information for replication by setting the error log level to **8192**, which is replication debugging. See Section 15.3.5, “Configuring Log Levels”.

To change the error log level to **8192** with **ldapmodify**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 8192
```

Because log level is additive, running the above command will result in excessive messages in the error log. So, use it judiciously.

To turn off replication debugging log, set the same attribute to **0**.

The **cl-dump.pl** script, which is explained in detail in the *Directory Server Configuration and Command-Line Tool Reference* can also help troubleshoot replication-related problems. Depending on the usage options, the script can selectively dump a particular replica:

- Dump the contents of a **replication-change-log** file and in-memory variables **purge RUV** and **maxRUV**.
- Grep and interpret change state numbers (CSNs) in the changelog.
- Get the base-64 encoded changelog from the Directory Server, and then decode the changelog.

Many common replication problems are described in Table 11.7, "Replication Errors".

Table 11.7. Replication Errors

Error/Symptom	Reason	Impact	Remedy
agmt=%s (%s:%d) Replica has a different generation ID than the local data	The consumer specified at the beginning of this message has not been (successfully) initialized yet, or it was initialized from a different root supplier.	The local supplier will not replicate any data to the consumer.	Ignore this message if it occurs before the consumer is initialized. Otherwise, reinitialize the consumer if the message is persistent. In a multi-master environment, all the servers should be initialized only once from a root supplier, directly or indirectly. For example, M1 initializes M2 and M4, M2 then initializes M3, and so on. The important thing to note is that M2 must not start initializing M3 until M2's own initialization is done (check the total update status from the M1's Console or M1 or M2's error log). Also, M2 should not initialize M1 back.
Warning: data for replica's was reloaded, and it no	This message may appear only when a supplier is restarted. It indicates that the supplier was unable to write the changelog or did not flush out its RUV at its last shutdown. The former is usually because of a disk-space problem, and the latter because a server crashed or was ungracefully shut down.	The server will not be able to send the changes to a consumer if the consumer's ***maxcsn*** no longer exists in the server's changelog.	Check the disk space and the possible core file (under the server's logs directory). If this is a single-master replication, reinitialize the consumers. Otherwise, if the server later complains that it can not locate some CSN for a consumer, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.
agmt=%s(%s:%d): Cannot locate CSN %s in the changelog (DB rc=%d). The consumer may need to be reinitialized.	Most likely the changelog was recreated because of the disk is full or the server ungracefully shutdown.	The local server will not be able to send any more change to that consumer until the consumer is reinitialized or gets the CSN from other suppliers.	If this is a single-master replication, reinitialize the consumers. Otherwise, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

Error/Symptom	Reason	Impact	Remedy
Too much time skew	The system clocks on the host machines are extremely out of sync.	The system clock is used to generate a part of the CSN. In order to reflect the change sequence among multiple suppliers, suppliers would forward-adjust their local clocks based on the remote clocks of the other suppliers. Because the adjustment is limited to a certain amount, any difference that exceeds the permitted limit will cause the replication session to be aborted.	Synchronize the system clocks on the Directory Server host machines. If applicable, run the network time protocol (**ntp**) daemon on those hosts.
agmt=%s(%s:%d): Warning: Unable to send endReplication extended operation (%s)	The consumer is not responding.	If the consumer recovers without being restarted, there is a chance that the replica on the consumer will be locked forever if it did not receive the release lock message from the supplier.	Watch if the consumer can receive any new change from any of its suppliers, or start the replication monitor, and see if all the suppliers of this consumer warn that the replica is busy. If the replica appears to be locked forever and no supplier can get in, restart the consumer.

Error/Symptom	Reason	Impact	Remedy

Changelog is getting too big.	Either changelog purge is turned off, which is the default setting, or changelog purge is turned on, but some consumers are way behind the supplier.		By default changelog purge is turned off. To turn it on from the command line, run **ldapmodify** as follows:
			
			`ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x`
			
			`dn: cn=changelog5,cn=config`
			`changetype: modify`
			`add: nsslapd-changelogmaxage`
			`nsslapd-changelogmaxage: 1d`
			
			where **1d** means 1 day. Other valid time units are **s** for seconds, **m** for minutes, **h** for hours, and **w** for weeks. A value of **0** turns off the purge.
			
			With changelog purge turned on, a purge thread that wakes up every five minutes will remove a change if its age is greater than the value of ***nsslapd-changelogmaxage*** and if it has been replayed to all the direct consumers of this supplier (supplier or hub).
			
			If it appears that the changelog is not purged when the purge threshold is reached, check the maximum time lag from the replication monitor among all the consumers. Irrespective of what the purge threshold is, no change will be purged before it is replayed by all the consumers.

The Replication Monitor is not responding. (For information on Replication Monitor, see Section 11.22, "Monitoring Replication Status".)	The SSL port is specified in some replication agreement, but the certificate database is not specified or not accessible by the Replication Monitor. If there is no SSL port problem, one of the servers in the replication topology might hang.		Map the SSL port to a non-SSL port in the configuration file of the Replication Monitor. For example, if 636 is the SSL port and 389 is the non-SSL port, add the following line in the **[connection]** section: `*:636=389:*:password`
In the Replication Monitor, some consumers show just the header of the table. (For information on Replication Monitor, see Section 11.22, "Monitoring Replication Status".)	No change has originated from the corresponding suppliers. In this case, the **MaxCSN:** in the header part should be **"None"**.		There is nothing wrong if there is no change originated from a supplier.