Administrator's Guide

Red Hat Directory Server

Version 7.1

May 2005

Red Hat, Inc.
1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588

Research Triangle Park NC 27709 USA

Contents

List of Figures

List of Tables

Introduction to This Reference Guide

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.

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 Class of Service — Provides a flexible mechanism for grouping and sharing attributes between entries in a dynamic fashion.
- Improved access control mechanism — 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 — Gives you 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 — Allows you to define a set of rules that govern how passwords and user accounts are managed in the Directory Server.
- SSL — Provides secure communications over the network including ciphers with up to 168-bit encryption.

The major components of Directory Server include:

- An LDAP server — The core of the directory service, provided by the `ns-slapd` daemon and compliant with the LDAP v3 Internet standards.
- Directory Server Console — An improved management console that dramatically reduces the effort of setting up and maintaining your directory service. The directory console is part of Red Hat Console, the common management framework for LDAP directory services.
- SNMP Agent — Permits you to monitor your Directory Server in real time using the Simple Network Management Protocol (SNMP).
- Online backup and restore — Allows you to create backups and restore from backups while the server is running.

Prerequisite Reading

This manual describes how to administer the Directory Server and its contents. However, this manual does not describe many of the basic directory and architectural concepts that you need to deploy, install, and administer your directory service successfully. Those concepts are contained in the *Red Hat Directory Server Deployment Guide*. You should read that book before continuing with this manual.

When you are familiar with Directory Server concepts and have done some preliminary planning for your directory service, you can install the Directory Server. The instructions for installing the various Directory Server components are contained in the *Red Hat Directory Server Installation Guide*.

Also, *Managing Servers with Red Hat Console* contains general background information on how to use the Red Hat Console. You should read and understand the concepts in that book before you attempt to administer Directory Server.

Conventions Used in This Book

This section explains the conventions used in this book.

- `Monospaced font`—This typeface is used for any text that appears on the computer screen or text that you should type. It is also used for filenames, functions, and examples.
-

NOTE	Notes and Cautions mark important information. Make sure you read the information before continuing with a task.

- The greater than symbol (>) is used as a separator for successive menu selections. For example, Object > New > User means that you should pull down the Object menu, drag the mouse down to highlight New, and drag the mouse across to the New submenu in which you must select User.
- Throughout this book you will see path references of the form:

  serverRoot`/slapd-`*serverID*`/...`

  serverRoot is the installation directory. The default installation directory is `/opt/redhat-ds/servers`. If you have installed Directory Server in a different location, you should adapt the path accordingly.

  serverID is the ID or identifier you assigned to an instance of Directory Server when you installed it. For example, if you gave the server an identifier of `phonebook`, then the actual path would look like this:

  ```
  /opt/redhat-ds/servers/slapd-phonebook/. . .
  ```

- In examples/sample code, paths assume that the Directory Server is installed in the default location `/opt/redhat-ds/servers`. If you have installed your Directory Server in a different location, adapt the paths accordingly. Also, all examples use `phonebook` for the server identifier where appropriate.

Related Information

The document set for Directory Server also contains the following guides:

- *Red Hat Directory Server Deployment Guide* — Provides an overview for planning your deployment of the Directory Server. Includes deployment examples.
- *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 Configuration, Command, and File Reference* — Provides reference information on the command-line scripts, configuration attributes, and log files shipped with Directory Server.
- *Red Hat Directory Server Schema Reference* — Provides reference information about the Red Hat Directory Server schema.
- *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.
- *Red Hat Directory Server Gateway Customization Guide* — Introduces Directory Server Gateway and explains how to implement a gateway instance with basic directory look-up functionality. Also contains information useful for implementing a more powerful gateway instance with directory authentication and administration capability.
- *Red Hat Directory Server Org Chart* — Introduces the Red Hat Directory Server Org Chart application and explains how to integrate it with an instance of Directory Server.
- *Red Hat Directory Server DSML Gateway Guide* — Introduces the Red Hat Directory Server DSML Gateway function and explains how to customize it for use as an independent gateway.

For a list of documentation installed with Directory Server, open this file:

serverRoot`/manual/en/slapd/index.htm`

For the latest information about Directory Server, including current release notes, complete product documentation, technical notes, and deployment information, check this site:

```
http://www.redhat.com/docs/manuals/dir-server/
```

Part 1

Administering Red Hat Directory Server

Chapter 11, “Managing SSL and SASL”

Chapter 12, “Monitoring Server and Database Activity”

Chapter 13, “Monitoring Directory Server Using SNMP”

Chapter 14, “Tuning Directory Server Performance”

Chapter 1

Introduction to Red Hat Directory Server

Red Hat Directory Server (Directory Server) product includes a Directory Server, an Administration Server to manage multiple server instances, and Red Hat Console to manage server instances through a graphical interface. This chapter provides overview information about the Directory Server and the most basic tasks you need to start administering a directory service.

It includes the following sections:

Overview of Directory Server Management

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). The Directory Server runs as the `ns-slapd` process or service on your machine. The server manages the directory databases and responds to client requests.

You perform most Directory Server administrative tasks through the Red Hat Administration Server, a second server that helps manage Directory Server. For Directory Server, you use a part of the Red Hat Administration Server called Red Hat Console. The Directory Server Console is a part of Red Hat Console designed specifically for use with Directory Server.

You can perform most Directory Server administrative tasks from the Directory Server Console. You can also perform administrative tasks manually by editing the configuration files or by using command-line utilities. For more information about the Red Hat Console, see *Managing Servers with Red Hat Console*.

Using the Directory Server Console

The Directory Server Console is an integral part of the Red Hat Console. You start the Directory Server Console from Red Hat Console, as described below.

Starting Directory Server Console

1. Check that the Directory Server daemon, `slapd-`*serverID*, is running. If it is not, as `root` user, enter the following command to start it:

   serverRoot`/slapd-`*serverID*`/start-slapd`

2. Check that the Administration Server daemon, `admin-serv`, is running. If it is not, as `root` user, enter the following command to start it:

   serverRoot`/start-admin`

3. Start Red Hat Console by entering the following command:

   serverRoot`/startconsole`

   The Console login window is displayed. If your configuration directory (the directory that contains the `o=NetscapeRoot` suffix) is stored in a separate instance of Directory Server, a window is displayed requesting the administrator user ID, password, and the URL of the Red Hat Administration Server for that Directory Server.

4. Log in using the bind DN and password of a user with sufficient access permissions for the operations you want to perform.

   For example, use `cn=Directory Manager` and the appropriate password. The Red Hat Console is displayed.

5. Navigate through the tree in the left-hand pane to find the machine hosting your Directory Server, and click on its name or icon to display its general properties.
6. To edit the name and description of your Directory Server, click the Edit button. Enter the new name and description in the text boxes. The name will appear in the tree on the left. Click OK to set the new name and description.
7. Double-click the name of your Directory Server in the tree or click the Open button to display the Directory Server Console.

Copying Entry DNs to the Clipboard

Using the Directory tab, you can copy the DN of an entry to the clipboard.

To do this:

1. In the Directory Server Console, select the Directory tab.
2. Browse through the tree until the entry whose DN you want to copy is displayed.
3. Select the entry in the tree, and then select Edit >Copy DN, or press Shift+Ctrl+C.

Configuring the Directory Manager

The *Directory Manager* is the privileged database administrator, comparable to the `root` user in UNIX. Access control does not apply to the entry you define as Directory Manager. You initially defined this entry during installation. The default is `cn=Directory Manager`.

The password for this user is defined in the `nsslapd-rootdn` attribute.

To change the Directory Manager DN and password and the encryption scheme used for this password:

1. Log in to the Directory Console as Directory Manager.

   If you are already logged in to the Console, see “Binding to the Directory from Red Hat Console,” on page 36, for instructions on how to log in as a different user.
2. In the Directory Server Console, select the Configuration tab, and then select the top entry in the navigation tree in the left pane.

3. Select the Manager tab in the right pane.
4. Enter the new distinguished name for the Directory Manager in the Root DN field.

   The default value is `cn=Directory Manager`.
5. From the Manager Password Encryption pull-down menu, select the storage scheme you want the server to use to store the password for Directory Manager.
6. Enter the new password, and confirm it using the text fields provided.
7. Click Save.

Binding to the Directory from Red Hat Console

When you create or manage entries from the Directory Server Console and when you first access the Red Hat Console, you are given the option to log in by providing a bind DN and a password. This option lets you indicate who is accessing the directory tree. This determines the access permissions granted to you and whether you can perform the requested operation.

Changing Login Identity

You can log in with the Directory Manager DN when you first start the Red Hat Console. At any time, you can choose to log in as a different user, without having to stop and restart the Console.

To change your login in Red Hat Console:

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

   A login dialog box appears.
3. Enter the new DN and password, and click OK.

   Enter the full distinguished name of the entry with which you want to bind to the server. For example, if you want to bind as the Directory Manager, then enter the following in the Distinguished Name text box:

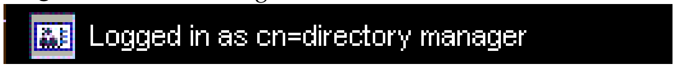
   ```
   cn=Directory Manager
   ```

For more information about the Directory Manager DN and password, refer to “Configuring the Directory Manager,” on page 35.

Viewing the Current Bind DN from the Console

You can view the bind DN you used to log in to the Directory Server Console by clicking the login icon in the lower-left corner of the display. The current bind DN appears next to the login icon, as shown here:

Figure 1-1 Viewing the Bind DN



Logged in as cn=directory manager

Starting and Stopping the Directory Server

Your Directory Server is running when you finish installing. It is best to avoid stopping and starting the server to prevent interrupting replication, searches, and other server operations.

- If you have enabled SSL for the Directory Server, 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 “Creating a Password File,” on page 439, for more information.
- Rebooting the system does not automatically start the `ns-slapd` process. This is because the directory does not automatically create startup or run command (`rc`) scripts. Check your operating system documentation for details on adding these scripts.
- If the Directory Server shuts down due to a full disk, subsequent restart of the server may take a very long time, even more than an hour. Ensure that the machine on which you install the server has adequate disk space and that the machine is configured appropriately to handle large files. For more information on setting these parameters, see chapter 3, “Computer System Requirements,” in the *Red Hat Directory Server Installation Guide*.

Starting and Stopping the Server from the Console

1. Start the Directory Server Console.

   For instructions, refer to "Starting Directory Server Console," on page 34.

2. In the Tasks tab, click "Start the Directory Server" or "Stop the Directory Server" as appropriate.

When you successfully start or stop your Directory Server from the Directory Server Console, the server displays a message box stating that the server has either started or shut down.

Starting and Stopping the Server from the Command-Line

Use one of the following scripts:

serverRoot`/slapd-`*serverID*`/start-slapd`

or

serverRoot`/slapd-`*serverID*`/stop-slapd`

where *serverID* is the identifier you specified for the server when you installed it.

Both of these scripts must run with the same UID and GID as the Directory Server. For example, if the Directory Server runs as `nobody`, you must run the `start-slapd` and `stop-slapd` utilities as `nobody`.

Configuring LDAP Parameters

You can view and change the parameters relevant to the server's network and LDAP settings through the Directory Server Console. This section provides information on:

- Changing Directory Server Port Numbers
- Placing the Entire Directory Server in Read-Only Mode
- Tracking Modifications to Directory Entries

For information on schema checking, see chapter 9, "Extending the Directory Schema."

Changing Directory Server Port Numbers

You can modify the port or secure port number of your user Directory Server using the Directory Server Console or by changing the value of the `nsslapd-port` attribute under the `cn=config` entry.

If you want to modify the port or secure port for a Directory Server that contains the configuration information (`o=NetscapeRoot` subtree), you may do so through Directory Server Console.

If you change the configuration directory or user directory port or secure port numbers, you should be aware of the following repercussions:

- You need to change the configuration or user directory port or secure port number configured for Red Hat Administration Server. See *Managing Servers with Red Hat Console* for information.
- If you have other Directory Servers installed that point to the configuration or user directory, you need to update those servers to point to the new port number.

To modify the port or secure port on which either a user or a configuration directory listens for incoming requests:

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. Enter the port number you want the server to use for non-SSL communications in the "Port" text box.

   The default value is `389`.
4. Enter the port number you want the server to use for SSL communications in the Encrypted Port text box.

   The encrypted port number that you specify must not be the same port number as you are using for normal LDAP communications. The default value is `636`.

5. Click Save.

   A warning will appear: "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 on Yes.

   Then a dialog will appear, saying 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 Administration Server.

6. Open the Administration Server Console.

   Open the Configuration tab, then select the Configuration DS tab.

7. In the "LDAP Port" field, type in the new LDAP port number for your Directory Server port.

   Check the "Secure Connection" box if this is a secure port.

NOTE If you try to save these changes at this step, you will get a warning box that reads, "Invalid LDAP Host/LDAP Port, can not connect." Click OK, and ignore this warning.

8. In the Task tab of the Directory Server Console, click on "Restart Directory Server." A dialog will appear, asking if you want to restart the server. Click Yes.

   See "Starting and Stopping the Directory Server," on page 37, for information.

9. Now you can go to the Configuration DS tab of the Administration Console and select Save.

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

10. In the Tasks tab of the Administration Server Console, click on "Restart Admin Server." A dialog will appear, saying that the Admin Server has been successfully restarted. Click on 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."

Placing the Entire Directory Server in Read-Only Mode

If you maintain more than one database with your Directory Server and you need to place all your databases in read-only mode, you can do this in a single operation. However, if your Directory Server contains replicas, you must 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 checkbox.
4. Click Save, and then restart the server.

NOTE 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 you have enabled read-only mode, you cannot undo it from the Console; you must modify the configuration files.

For information on placing a single database in read-only mode, refer to "Enabling Read-Only Mode," on page 168.

Tracking Modifications to Directory Entries

You can configure the server to maintain special attributes for newly created or modified entries:

- `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.

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. For information on proxy authorization, refer to "Providing Bind Credentials," on page 113.

To enable the Directory Server to track this information:

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 checkbox.

   The server adds the `creatorsName`, `createTimestamp`, `modifiersName`, and `modifyTimestamp` attributes to every newly created or modified entry.
4. Click Save, and then restart the server.

   See "Starting and Stopping the Directory Server," on page 37, for more information.

Cloning a Directory Server

Once you have set up and configured your Directory Server, Red Hat Console offers a simple way of duplicating your configuration on another instance of the Directory Server. This is a two-phase procedure:

- First, you must create a new instance of the Directory Server.
- Second, you must clone the configuration of your first Directory Server instance and apply it to the new one.

NOTE The configuration information that is duplicated during these operations does not include the `o=NetscapeRoot` suffix of the configuration directory.

Creating a New Directory Server Instance

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 of > Directory Server.

   The Create New Instance dialog box is displayed.
3. Enter a unique identifier for the server in the Server Identifier field.

   This name must not include the period (.) symbol.
4. Enter the a port number for LDAP communications in the Network port field.
5. Enter the suffix managed by this new instance of the directory in the base suffix field.
6. Enter a DN for the Directory Manager in the Root DN field.

   For information on the role and privileges of the Directory Manager entry, refer to “Configuring the Directory Manager,” on page 35.
7. Enter the password for this user in the Password for Root DN field, and confirm it by entering it again in the Confirm Password field.
8. Enter the user ID for the Directory Server daemon in the Server Runtime User ID field.

9. Click OK.

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

Cloning the Directory Configuration

1. In the Red Hat Console window, expand the Server Group folder, and right-click on the Directory Server that you want to clone.
2. From the pop-up menu, select Clone Server Config.

   A new window is displayed with the list of target servers for cloning.
3. In this window, select the server to which you want the configuration to apply, and click the Clone To button.

   A message is displayed to give you the status of the operation.

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, you can also start Directory Server in referral mode if you're making configuration changes to the Directory Server and you want all clients to be referred to another supplier for the duration. To do this, you must start the server with the `refer` command.

If the server is already running, you can put it in referral mode by using the Directory Server Console. This procedure is explained in "Setting Default Referrals," on page 144.

Using the refer Command

Follow these steps to start the Directory Server in referral mode:

1. Go to the `/bin/slapd/server` directory under your installation directory:

   ```
   cd serverRoot/slapd-serverID/bin/slapd/server
   ```

2. Run the `refer` command, as follows:

```
./ns-slapd refer -D instance_dir [-p port] -r referral_url
```

instance_dir is the directory instance for which queries will be referred, *port* is the option port number of the Directory Server you want to start in referral mode, and *referral_url* is the referral returned to clients. For information on the format of an LDAP URL, refer to Appendix C, "LDAP URLs."

Chapter 2

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.

During the planning phase of your directory deployment, you should characterize the types of data that your directory will contain. You should read *Red Hat Directory Server Deployment Guide* before creating entries and modifying the default schema. Also, entries stored in an Active Directory or Windows NT4 directory service can be added to the Directory Server through Windows User Sync; see chapter 18, “Windows Sync,” for more information on adding or modifying synchronized entries through Windows User Sync.

This chapter consists of the following sections:

- Managing Entries from the Directory Console (page 47)
- Managing Entries from the Command-Line (page 57)
- LDIF Update Statements (page 65)
- Maintaining Referential Integrity (page 75)

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.

If you want to add several entries simultaneously, you can use the command-line utilities described in “Managing Entries from the Command-Line,” on page 57.

This section provides information about:

- Creating a Root Entry

- Creating Directory Entries
- Modifying Directory Entries
- Deleting Directory Entries

This section assumes some basic knowledge of object classes and attributes. For an introduction to object classes and attributes, refer to *Red Hat Directory Server Deployment Guide*. For information on the definition and use of all schema provided with the Directory Server, refer to the *Red Hat Directory Server Schema Reference*.

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 6, "Managing Access Control."

Creating a Root Entry

Each time you create a new database, you associate it 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.

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Create a new database, as explained in "Creating and Maintaining Databases," on page 92.

3. 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 a list of suffixes that do not have a corresponding entry.

4. Choose the suffix corresponding to the entry that you want to create.

   The New Object window is displayed.

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

   The object class you select must contain the attribute you used to name the suffix. For example, if you are creating the entry corresponding to the suffix `ou=people,dc=example,dc=com`, then you can 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 is displayed. You can either accept the current values by clicking OK or modify the entry, as explained in "Modifying Directory Entries," on page 51.

Creating Directory Entries

Directory Server Console offers several predefined templates for creating directory entries. Templates are available for the following types of entries:

- User
- Group
- Organizational Unit
- Role
- Class of Service

Table 2-1 shows what type of object class is used for each template.

Table 2-1 Entry Templates and Corresponding Object Classes

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

These templates contain fields representing all the mandatory attributes and some of the commonly used optional attributes. To create an entry using one of these templates, refer to "Creating an Entry Using a Predefined Template," on page 50. To create any other type of entry, refer to "Creating Other Types of Entries," on page 50.

Creating an Entry Using a Predefined Template

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Right-click the entry in the left pane under which you want to add the new entry, and select the appropriate type of entry: User, Group, Organizational Unit, Role, Class of Service, or Other.

   The corresponding Create window is displayed.

3. Supply values for all of the mandatory attributes (identified by an asterisk) and, if you want, for any of the optional attributes.

   The Create window does not provide fields for all optional attributes.

4. To display the full list of attributes, click the Advanced button.

   The Property Editor is displayed. Refer to "Modifying Directory Entries," on page 51, for information on using the Property Editor.

5. Click OK to dismiss the Create window.

   The new entry is displayed in the right pane.

Creating Other Types of Entries

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Right-click the entry in the left pane under which you want to add the new entry, and select Other.

   The New Object window is displayed.

3. In the object class list, select an object class that defines your new entry.

4. Click OK.

   If you selected an object class related to a type of entry for which a predefined template is available, the corresponding Create window is displayed. (See "Creating an Entry Using a Predefined Template," on page 50).

   In all other cases, the Property Editor is displayed. It contains a list of mandatory attributes.

5. Supply a value for all the listed attributes.

   Some fields are empty, but some might have a generic placeholder value (such as `New`), which you should replace with a meaningful value for your entry.

   Some object classes can have several naming attributes. Remember to select the naming attribute you want to use to name your new entry.

   To provide values for optional attributes that are not listed, refer to "Modifying Directory Entries," on page 51.

6. Click OK to save the new entry and dismiss the Property Editor window.

   The new entry is displayed in the right pane.

Modifying Directory Entries

To modify directory entries from Directory Server Console, you must start the Property Editor. The Property Editor contains the list of object classes and attributes belonging to an entry.

From the Property Editor, you can:

- Add an object class to an entry.
- Remove an object class from an entry.
- Add an attribute to an entry.
- Add an attribute value to an entry.
- Remove an attribute value from an entry.
- Add an attribute subtype to an entry.

This section describes how to start the Property Editor and how to use it to modify an entry's attributes and attribute values.

Displaying the Property Editor

You can start the Property Editor in several ways:

- From the Directory tab, by right-clicking an entry in the left or right pane, and selecting Properties from the pop-up menu.
- From the Directory tab, by double-clicking an entry in the left or right pane.
- From the Create User, Group, Organizational Unit, Role, and Class of Service templates, by clicking the Advanced button (see "Creating an Entry Using a Predefined Template," on page 50).
- From the New Object window, by clicking OK (see "Creating Other Types of Entries," on page 50).

Adding 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 you want to modify, and select Advanced from the pop-up menu.

   You can also double-click the entry. The Property Editor is displayed; click on the Advanced button.

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

   The Add Object Class window is displayed. It shows a list of object classes that you can add to the entry.

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

   The object class you selected appears in the list of object classes in the Advanced Property Editor. To dismiss the Add Object Class window, click Cancel.

4. Click OK in the Advanced Property Editor when you have finished editing the entry. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

Removing an Object Class

To remove an object class from an entry:

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

   You can also double-click the entry. The Property Editor is displayed; click on the Advanced button.

2. Click the cursor in the text box that shows the object class you want to remove, and click Delete Value.

3. Click OK in the Advanced Property Editor when you have finished editing the entry. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

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 "Adding an Object Class to an Entry," on page 52, and chapter 9, "Extending 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 you want to modify, and select Advanced from the pop-up menu.

   You can also double-click the entry. The Property Editor is displayed; click on the Advanced button.

2. Click Add Attribute.

   The Add Attribute dialog box is displayed.

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

   The Add Attribute window is dismissed, and the attribute you selected appears in the list of attributes in the Advanced Property Editor.

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

5. Click OK in the Advanced Property Editor when you have finished editing the entry. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

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 2Mbytes. LDAP add or modify operations will fail when attempting to add very large attributes that result in a request that is larger than 2Mbytes.

To add very large attributes, you must 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, you must consider all elements of the LDAP add and modify operations used to add the attributes, not just the single attribute. The list of what is included in determining this size is as follows:

- 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 (10Kbytes should be sufficient).

For further information about the `nsslapd-maxbersize` attribute and for information about setting this attribute, see the section "nsslapd-maxbersize (Maximum Message Size)" in chapter 2, "Core Server Configuration Reference," in *Red Hat Directory Server Configuration, Command, and File Reference*.

Adding Attribute Values

When an entry contains multi-valued attributes, you can supply multiple values for these attributes.

To add an attribute value to a multi-valued attribute:

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

   You can also double-click the entry. The Property Editor is displayed; click on the Advanced button.

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

   A new blank text field is displayed in the right column.

3. Type in the name of the new attribute value.

4. Click OK in the Advanced Property Editor when you have finished editing the entry. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

Removing an Attribute Value

To remove an attribute value from an entry:

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

   You can also double-click the entry. The Property Editor is displayed; click on the Advanced button.

2. Click the cursor in the text box that contains the attribute value you want to remove, and click Delete Value.

   If you want to remove the entire attribute and all its values from the entry, select Delete Attribute from the Edit menu.

3. Click OK in the Advanced Property Editor when you have finished editing the entry. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

Adding an Attribute Subtype

You can add three different kinds of subtypes to attributes contained within an entry: language, binary, and pronunciation.

Language Subtype

Sometimes a user's name can be more accurately represented in characters of a language other than the default language. For example, Noriko's name is Japanese, and she 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 Japanese name.

If you specify a language subtype for an attribute, the subtype is added to the attribute name as follows:

attribute`;lang-`*subtype*

attribute is the attribute you are adding to the entry and *subtype* is the two character abbreviation for the language. See Table D-2, on page 619, for a list of supported language subtypes. For example:

```
givenname;lang-ja
```

You can assign only one language subtype 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:Smith
```

Instead, use:

```
cn: lang-ja: ja_value
cn: lang-en-GB: en-GB_value
```

Binary Subtype

Assigning the binary subtype to an attribute indicates that the attribute value is binary. For example, `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.

You might want to use this 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 entry's Japanese name.

To Add a Subtype Using the Property Editor

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

   You can also double-click the entry. The Property Editor is displayed.

2. Click Add Attribute.

   The Add Attribute dialog box displays.

3. Select the attribute you want to add from the list.

4. To assign a language subtype to the attribute, select it from the Language drop-down list.

5. From the Subtype drop-down list, you can also assign one of two other subtypes, binary or pronunciation.
6. Click OK.

   The Add Attribute window is dismissed.
7. When you have finished defining the information for the entry, click OK in the Advanced Property Editor. The Advanced Property Editor is dismissed.

   Click OK in the Property Editor. The Property Editor is dismissed.

Deleting Directory Entries

To delete entries using the Directory Server Console:

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.
2. Right-click the entry you want to delete in the navigation tree or in the right pane, and select Delete from the pop-up menu.

   To select multiple entries, use Ctrl+click or Shift+click, and then select Delete from the Edit menu.

   The server deletes the entry or entries immediately. There is no undo.

Managing Entries from the Command-Line

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

Using the command-line utilities, you can provide information directly from the command-line or through an input file in LDIF.

This section provides information about:

- Providing Input from the Command-Line
- Creating a Root Entry from the Command-Line

- Adding Entries Using LDIF
- Adding and Modifying Entries Using ldapmodify
- Deleting Entries Using ldapdelete
- 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 your directory, see chapter 6, "Managing Access Control."

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, refer to "LDIF Update Statements," on page 65.

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 you finish providing input, enter the character that your shell recognizes as the end of file (EOF) escape sequence. The utility then begins operations based on the input you supplied.

Typically, the EOF escape sequence is one of the following, depending upon the type of machine you use, almost always control-D (`^D`).

For example, suppose you want to input some LDIF update statements to `ldapmodify`. Then, you would do the following:

```
prompt> ldapmodify -D bindDN -w password -h hostname
> 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
prompt>
```

When you add 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, if you want to place an entry in a `People` subtree, then 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.
...
```

Creating a Root Entry from the Command-Line

You can use the `ldapmodify` command-line utility to create a new root entry in a database. For example, you might add the new root entry as follows:

```
prompt> ldapmodify -a -D bindDN -w password
```

The `ldapmodify` utility binds to the server and prepares it to add an entry.

You create the new root object as follows:

```
dn: Suffix_Name
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 mandatory attributes depending upon the root object you add.

NOTE You can use this method 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 `-n` option to specify the database that will hold the new entries. For information, refer to "Importing from the Command-Line," on page 155.

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 is described in Appendix A, "LDAP Data Interchange Format."

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

   See "Importing a Database from the Console," on page 152, for information. When you import the LDIF file, select "Append to database" on 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.

Adding and Modifying Entries Using ldapmodify

You use the `ldapmodify` command to add and modify entries in an existing Directory Server database. The `ldapmodify` command opens a connection to the specified server using the distinguished name and password you supply 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.

If schema checking is turned on when you use this utility, then the server performs schema checking for the entire entry when it is modified:

- 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. This situation can occur if you run the Directory Server with schema checking turned off, add unknown object classes or attributes, and then turn schema checking on.

For more information, see "Turning Schema Checking On and Off," on page 389.

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

Adding Entries Using ldapmodify

Here is a typical example of how to use the `ldapmodify` utility to add entries to the directory. Suppose that:

- You want to create the entries specified in the file `new.ldif`.
- You have created a database administrator who has the authority to modify the entries and whose distinguished name is `cn=Directory Manager, dc=example,dc=com`.
- The database administrator's password is `King-Pin`.
- The server is located on `cyclops`.
- The server uses port number `845`.

In this example, the LDIF statements in the `new.ldif` file do not specify a change type. They follow the format defined in "LDIF File Format," on page 575.

To add the entries, you must enter the following command:

```
ldapmodify -a -D "cn=Directory Manager,dc=example,dc=com" -w
King-Pin -h cyclops -p 845 -f new.ldif
```

The following table describes the `ldapmodify` parameters used in the example:

Table 2-2 Description of 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.
`-p`	Specifies the port number that the server uses.

Table 2-2 Description of ldapmodify Parameters Used for Adding Entries *(Continued)*

Parameter Name	Description
`-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, refer to "Providing Input from the Command-Line," on page 58.

For full information on `ldapmodify` parameters, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

Modifying Entries Using ldapmodify

Here is a typical example of how to use the `ldapmodify` utility to modify entries that are present in the directory. Suppose that:

- You want to modify entries as specified in the file `modify_statements`.
- You have created a database administrator that has the authority to modify the entries and whose distinguished name is `cn=Directory Manager, dc=example,dc=com`.
- The database administrator's password is `King-Pin`.
- The server is located on `cyclops`.
- The server uses port number `845`.

To modify the entries, you must first create the `modify_statements` file with the appropriate LDIF update statements and then enter the following command:

```
ldapmodify -D "cn=Directory Manager,dc=example,dc=com" -w
King-Pin -h cyclops -p 845 -f modify_statements
```

The following table describes the `ldapmodify` parameters used in the example:

Table 2-3 Description of 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.

Table 2-3 Description of ldapmodify Parameters Used for Modifying Entries

Parameter Name	Description
`-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, refer to "Providing Input from the Command-Line," on page 58.

For full information on `ldapmodify` parameters, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

Deleting Entries Using ldapdelete

Use the `ldapdelete` command-line utility to delete entries from the directory. This utility opens a connection to the specified server using the distinguished name and password you provide and deletes the entry or entries.

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:

```
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
```

you can delete only the last two entries. The entry that identifies the `People` subtree can be deleted only if there aren't any entries below it. If you want 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.

Here is a typical example of how to use the `ldapdelete` utility. Suppose that:

- You want to delete the entries identified by the distinguished names `cn=Robert Jenkins,ou=People,dc=example,dc=com` and `cn=Lisa Jangles, ou=People,dc=example,dc=com`.

- You have created a database administrator that has the authority to modify the entries, and whose distinguished name is `cn=Directory Manager, dc=example,dc=com`.
- The database administrator's password is `King-Pin`.
- The server is located on `cyclops`.
- The server uses port number `845`.

To delete the entries for users Robert Jenkins and Lisa Jangles, enter the following command:

```
ldapdelete -D "cn=Directory Manager,dc=example,dc=com" -w
King-Pin -h cyclops -p 845 "cn=Robert
Jenkins,ou=People,dc=example,dc=com" "cn=Lisa
Jangles,ou=People,dc=example,dc=com"
```

The following table describes the `ldapdelete` parameters used in the example:

Table 2-4 Description of 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.

For full information on `ldapdelete` parameters, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

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 [*], backslash [\], and so forth). 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 you use, you should use either single or double quotation marks for this purpose. Refer to your operating system documentation for more information.

In addition, if you are using DNs that contain 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, you would enter the following command:

```
ldapdelete -D "cn=Directory Manager,dc=example,dc=com" -w
King-Pin -h cyclops -p 845 "cn=Patricia
Fuentes,ou=People,o=example.com Bolivia\,S.A."
```

LDIF Update Statements

Use LDIF update statements to define how `ldapmodify` should change your directory. In general, LDIF update statements are a series of statements that:

- Specify the distinguished name of the entry to be modified.
- Specify a change type that defines how a specific entry is to be modified (`add`, `delete`, `modify`, `modrdn`).
- Specify a series of attributes and their changed values.

A change type is required unless you use `ldapmodify` 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_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.

Adding an Entry Using LDIF

Use `changetype: add` to add an entry to your 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, if you want 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.

The following LDIF update statements can be used to create the `People` and the `Groups` subtrees and then to create entries within those trees:

```
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
```

Renaming an Entry Using LDIF

Use `changetype:modrdn` to change an entry's relative distinguished name (RDN). An entry's RDN is the left-most element in the distinguished name. Therefore, 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
```

Therefore, this rename operation allows you to change the left-most value in an entry's distinguished name.

For example, the entry

```
cn=Sue Jacobs,ou=People,dc=example,dc=com
```

can be modified to be

```
cn=Susan Jacobs,ou=People,dc=example,dc=com
```

but it cannot be modified to be

```
cn=Sue Jacobs,ou=old employees,dc=example,dc=com
```

The following example can be used to rename 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, if you used

```
dn: cn=Sue Jacobs,ou=Marketing,dc=example,dc=com
changetype: modrdn
newrdn: cn=Susan Jacobs
deleteoldrdn: 1
```

the server would delete `cn=Sue Jacobs`, and only `cn=Susan Jacobs` would remain within the entry.

A Note on Renaming Entries

You cannot rename an entry with the `modrdn` change type such that the entry moves to a completely different subtree. To move an entry to a completely different branch, you must create a new entry in the alternative subtree using the old entry's attributes, and then delete the old entry.

Also, for the same reasons that you cannot delete an entry if it is a branch point, you cannot rename an entry if it has any children. Doing so would orphan the children in the tree, which is not allowed by the LDAP protocol. For example, of the following three entries:

```
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
```

you can rename only the last two entries. The entry that identifies the `People` subtree can be renamed only if no other entries exist below it.

Modifying an Entry Using LDIF

Use `changetype:modify` to add, replace, or remove attributes and/or attribute values to the 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 value(s). 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:

- Adding Attributes to Existing Entries Using LDIF
- Changing an Attribute Value Using LDIF
- Deleting All Values of an Attribute Using LDIF
- Deleting a Specific Attribute Value Using LDIF

Adding Attributes to Existing Entries Using LDIF

You use `changetype:modify` with the add operation to 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. The `jpeg` photo can be displayed by Directory Server Gateway. In order to add this attribute to the directory, you must use the `ldapmodify -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
```

If you use this standard notation, you do not need to specify the `ldapmodify -b` parameter. However, you must add the following line to the beginning of your LDIF file or your LDIF update statements:

```
version:1
```

For example, you could use the following `ldapmodify` command:

```
prompt> ldapmodify -D userDN -w user_password
```

```
>version: 1
>dn: cn=Barney Fife,ou=People,dc=example,dc=com
>changetype: modify
>add: userCertificate
>userCertificate;binary:< file: BarneysCert
```

NOTE You can use the standard LDIF notation only with the `ldapmodify` command, not with other command-line utilities.

Changing an Attribute Value Using LDIF

Use `changetype:modify` with the replace operation to change 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 that you want to change and then add the replacement value. 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 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
```

Barney's 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
```

Deleting All Values of an Attribute Using LDIF

Use `changetype:modify` with the delete operation to delete an attribute from an entry. If the entry has more than one instance of the attribute, you must indicate which of the attributes you want 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
```

If you want to delete just a specific instance of the `telephonenumber` attribute, then you simply delete that specific attribute value. The following section describes how to do this.

Deleting a Specific Attribute Value Using LDIF

Use `changetype:modify` with the delete operation to delete an attribute value from an entry.

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
```

Deleting an Entry Using LDIF

Use `changetype:delete` to delete an entry from your directory. 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:

```
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
```

you can delete only the last two entries. 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
```

CAUTION Do not delete the suffix `o=NetscapeRoot`. The Red Hat Administration Server uses this suffix to store information about installed Directory Servers. Deleting this suffix could force you to reinstall the Directory Server.

Modifying an Entry in an Internationalized Directory

If the attribute values in your directory are associated with one or more 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, if you want to modify an attribute value that has a language tag of `lang-fr`, you must 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
```

Maintaining Referential Integrity

Referential integrity is a database mechanism that ensures relationships between related entries are maintained. In the Directory Server, referential integrity can be used to ensure that an update to one entry in the directory is correctly reflected in any other entries that may refer 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.

How Referential Integrity Works

When the Referential Integrity Plug-in (see "Referential Integrity Postoperation Plug-in," on page 510) 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 your performance resource and time needs, as well as your integrity needs. Integrity checks can be time-consuming and draining on memory/CPU.

Whenever you delete or rename a user or group entry in the directory, the operation is logged to the referential integrity log file (*serverRoot*`/slapd-`*serverID*`/logs/referint`). After a specified time, known as the *update interval*, the server performs a search on all attributes for which referential integrity is enabled and matches the entries resulting from that search with the DNs of deleted or modified entries present in the log file. If the log file shows that the entry was deleted, the corresponding attribute is deleted. If the log file shows that the entry was changed, the corresponding attribute value is modified accordingly.

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. You can, however, configure the behavior of the Referential Integrity Plug-in to suit your own needs. You can do any of the following:

- Record referential integrity updates in the replication changelog.
- Modify the update interval.
- Select the attributes to which you apply referential integrity.
- Disable referential integrity.

Using Referential Integrity with Replication

There are certain limitations associated with the use of the Referential Integrity Plug-in in a replication environment:

- You should never enable it on a dedicated consumer server (a server that contains only read-only replicas).
- You should never enable it on a server that contains a combination of read-write and read-only replicas.
- You can enable it on a supplier server that contains only read-write replicas.

- In the context of multi-master replication, you should enable it on just one supplier.

Configuring the Supplier Server

When your replication environment satisfies the conditions listed above, you can enable the Referential Integrity Plug-in.

1. Enable the Referential Integrity Plug-in.

   This task is described in "Enabling/Disabling Referential Integrity," on page 77.

2. Configure the plug-in to record any integrity updates in the changelog.

   This task is described in "Recording Updates in the Changelog," on page 78.

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.

Enabling/Disabling Referential Integrity

You can enable or disable referential integrity from the Directory Server Console or from the command-line.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Expand the Plugins folder in the navigation tree, and select the Referential Integrity Postoperation Plug-in.

   The settings for the plug-in are displayed in the right pane.

3. Check the "Enable plugin" checkbox to enable the plug-in; clear it to disable it.

4. Click Save to save your changes.

5. For your changes to be taken into account, go to the Tasks tab, and select Restart the Directory Server.

Recording Updates in the Changelog

You can decide to record updates in the replication changelog instead of recording them in the default location, the `referint` file in the *serverRoot*`/slapd-`*serverID*`/logs` directory. You must do this if you want referential integrity updates to be replicated to consumer servers in the context of replication.

You can make this change from the Directory Server Console.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Expand the Plugins folder in the navigation tree, and select the Referential Integrity Postoperation Plug-in.

   The settings for the plug-in are displayed in the right pane.

3. In the arguments list, replace the `referint` filename with the absolute path to the changelog directory.
4. Click Save to save your changes.
5. For your changes to be taken into account, go to the Tasks tab, and select Restart the Directory Server.

Modifying the Update Interval

By default, the server makes referential integrity updates immediately after a delete or a modrdn operation. If you want to reduce the impact this operation has on your system, you may want to increase the amount of time between updates. Although there is no maximum update interval, the following intervals are commonly used:

- Update immediately.
- 90 seconds (updates occur every 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).

You can modify the update interval from the Directory Server Console.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 34.

2. Expand the Plugins folder in the navigation tree, and select the Referential Integrity Postoperation Plug-in.

   The settings for the plug-in are displayed in the right pane.

3. In the arguments list, replace the value in the first text box with the appropriate time interval.
4. Click Save to save your changes.
5. For your changes to be taken into account, go to the Tasks tab, and select Restart the Directory Server.

Modifying the Attribute List

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 from the Directory Server Console. For example, you may add the `nsroledn` attribute if roles are being used.

Keep in mind that any attribute specified in the Referential Integrity Plug-in parameter list needs to 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 backends.

You can improve the performance by removing any unused attributes from the list.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to “Using the Directory Server Console,” on page 34.

2. Expand the Plugins folder in the navigation tree, and select the Referential Integrity Postoperation Plug-in.

   The settings for the plug-in are displayed in the right pane.

3. In the Arguments section, use the Add and Delete buttons to modify the attributes in the list.

4. Click Save to save your changes.

5. For your changes to be taken into account, go to the Tasks tab, and select Restart the Directory Server.

NOTE For best performance, the attributes set for updating should also be indexed. For information on indexing, see chapter 10, “Managing Indexes.”

Chapter 3

Configuring Directory Databases

Your directory is made up of databases over which you can distribute your directory tree. This chapter describes how to create *suffixes*, the branch points for your 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.

This chapter includes the following sections:

- Creating and Maintaining Suffixes (page 81)
- Creating and Maintaining Databases (page 92)
- Creating and Maintaining Database Links (page 104)
- Using Referrals (page 144)

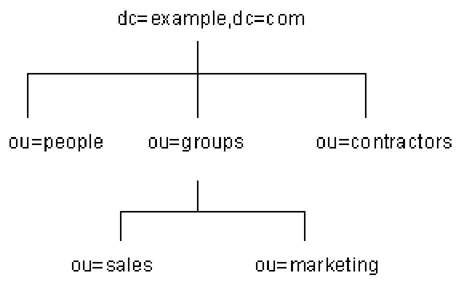
For conceptual information on distributing your directory data, refer to the *Red Hat Directory Server Deployment Guide*.

Creating and Maintaining Suffixes

You can store different pieces of your directory tree in different databases and then distribute these databases across multiple servers. Your directory tree contains branch points called nodes. These nodes may be associated with databases. You create nodes using the Directory tab in the Directory Server Console, where you freely edit the entries that appear in your directory tree.

A suffix is a node of your directory tree associated with a particular database. You create these special nodes using the Database tab on the Directory Server Console. For example, a simple directory tree might appear as illustrated in Figure 3-1.

Figure 3-1 A Sample 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.

This section describes creating suffixes on your Directory Server and associating them with databases. This section contains procedures for the following:

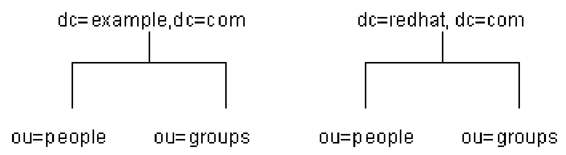
- Creating Suffixes
- Maintaining Suffixes

Creating Suffixes

You can create both root and subsuffixes to organize the contents of your directory tree. A root suffix is the parent of a sub suffix. It can be part of a larger tree you have designed for your Directory Server. A sub suffix is a branch underneath a root suffix. The data for root and subsuffixes are contained by databases.

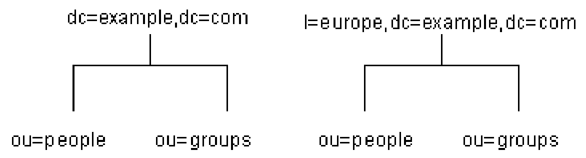
Your directory might contain more than one root suffix. For example, an ISP might host several websites, one for `example.com` and one for `redhat.com`. The ISP would create two root suffixes, one corresponding to the `dc=example,dc=com` naming context and one corresponding to the `dc=redhat,dc=com` naming context. The directory tree appears as illustrated in Figure 3-2.

Figure 3-2 A Sample Directory Tree with Two Root Suffixes



You can also create root suffixes to exclude portions of your directory tree from search operations. For example, `example.com` Corporation might want to exclude their European office from a search on the general `example.com` Corporation directory. To do this, they create two root suffixes. One root suffix corresponds to the general `example.com` 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 3-3.

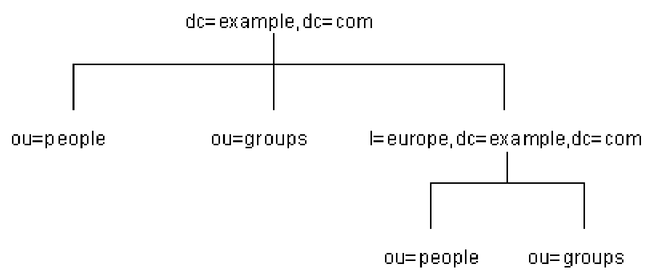
Figure 3-3 A Sample Directory Tree with a Root Suffix Off Limits to Search Operations



Searches performed by client applications on the `dc=example,dc=com` branch of `example.com` 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.com` Corporation decides that they want to include the entries in the European branch of their directory tree in general searches, they would make the European branch a sub suffix of the general branch. To do this, they create a root suffix `example.com` 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 would appear as illustrated in Figure 3-4.

Figure 3-4 A Sample Directory Tree with a Sub Suffix



This section describes creating root and subsuffixes for your directory using either the Directory Server Console or the command-line. This section contains the following procedures:

- Creating a New Root Suffix Using the Console
- Creating a New Sub Suffix Using the Console
- Creating Root and Sub Suffixes from the Command-Line

Creating a New Root Suffix Using the Console

The following procedure describes creating a suffix and associating it with a database:

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.

   The "Create new root suffix" dialog box is displayed.
3. Enter a unique suffix in the "New suffix" field.

   The suffix must be named according to dc naming conventions. For example, you might enter a new suffix name of `dc=example,dc=com`.
4. Select the "Create associated database automatically" checkbox if you want a database to be created at the same time as the new root suffix.

   Deselect the checkbox if you want to create a database for the new root suffix later. You may want to do this if you want to be able to specify a directory where the database will be created. The new root suffix will be disabled until you create a database.

5. If you selected the "Create associated database automatically" checkbox in step 4, enter a unique name for the new database in the "Database name" field.

   For the name, you can use a combination of alphanumeric, dash (`-`), and underscore (_) characters; no other characters are allowed. For example, you might name the new database `example2`.

6. Click OK to create the new root suffix.

   The suffix appears automatically under the Data branch in the left navigation pane.

Creating a New Sub Suffix Using the Console

The following procedure describes creating a sub suffix under an already existing root or sub suffix:

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

2. Under the Data in the left navigation pane, select the suffix under which you want 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 according to dc naming conventions. For example, you might enter a new suffix name of `ou=groups`.

   The root suffix is automatically added to the name. For example, if you are creating the sub suffix `ou=groups` 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" checkbox if you want a database to be created at the same time as the new sub suffix.

   Deselect the checkbox if you want to create a database for the new sub suffix later. The new sub suffix will be disabled until you create a database.

5. If you selected the "Create associated database automatically" checkbox in step 4, enter a unique name for the new database in the "Database name" field.

   For the name, you can use a combination of alphanumeric, dash (`-`), and underscore (_) characters; no other characters are allowed. For example, you might name the new database `example3`.

6. Click OK to create the new sub suffix.

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

Creating Root and Sub Suffixes from the Command-Line

Use the `ldapmodify` command-line utility to add new suffixes to your 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 probably suffer.

For example, you want to add a new root suffix to the configuration file using the `ldapmodify` utility. First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Then, run `ldapmodify`, as follows:

```
ldapmodify -a -h example1 -p 389 -D "cn=directory manager" -w
secret
```

The `ldapmodify` utility binds to the server and prepares it to add an entry to the configuration file.

Next, you create the root suffix entry for `example.com` Corporation, as follows:

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

To create a sub suffix for groups under this root suffix, you would do an `ldapmodify` operation to add the following entry:

```
dn: cn="ou=groups,dc=example,dc=com",cn=mapping tree,cn=config
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 If you want to maintain your suffixes using the Directory Server Console, you will need to respect the same spacing you use to name the root and subsuffixes via the command-line.

For example, if you name a root suffix `ou=groups  ,dc=example,dc=com` (with two spaces after `groups`), any subsuffixes you create 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 3-1 Suffix Attributes

Attribute Name	Value
`dn`	Defines the DN for the suffix. The DN is contained in quotes. The value you enter takes the following form: `cn="dc=domain,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.

Table 3-1 Suffix Attributes *(Continued)*

Attribute Name	Value
`nsslapd-state`	Determines how the suffix handles operations. This attribute takes the following values: • `backend`: the backend (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`.
`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. Refer to “Creating and Maintaining Database Links,” on page 104, 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 you have specified more than one database in the `nsslapd-backend` attribute. Refer to “Creating and Maintaining Databases,” on page 92, for more information about the custom distribution function.
`nsslapd-distribution-funct`	Specifies the name of your custom distribution function. This attribute is required only when you specify more than one database in the `nsslapd-backend` attribute. Refer to “Creating and Maintaining Databases,” on page 92, for more information about the custom distribution function.

Table 3-1 Suffix Attributes *(Continued)*

Attribute Name	Value
`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, you want to create a sub suffix `o=sales,dc=example,dc=com` under the root suffix `dc=example,dc=com`. Add the following value to the `nsslapd-parent-suffix` attribute of the sub suffix: `nsslapd-parent-suffix: "dc=example,dc=com"`

Maintaining Suffixes

This section describes the following procedures:

- Using Referrals in a Suffix
- Enabling Referrals Only During Update Operations
- Disabling a Suffix
- Deleting a Suffix

Using Referrals in a Suffix

Referrals can be used to point a client application temporarily to a different server. For example, you might add a referral to a suffix so that the suffix points to a different server when the database associated with the suffix is taken off-line for maintenance.

For more information on referrals in general, refer to *Red Hat Directory Server Deployment Guide*.

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 you want to add a referral.
3. Click the Suffix Settings tab, and select the “Return Referrals for all Operations” radio button.

4. Click the Referrals tab. Enter an LDAP URL in the “Enter a new referral” field, or click Construct to be guided through the creation of an LDAP URL.

   For more information about the structure of LDAP URLs, see Appendix C, “LDAP URLs.”

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

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

6. Click Save.

Enabling Referrals Only During Update Operations

You may want to configure your directory to redirect update and write requests made by client applications to a read-only database.

For example, you can enable referrals for update operations when you have a local copy of the directory data that you do not own. You want the data to be available for searches but not for updates. You do this by enabling referrals only during update requests. When a client application asks to update an entry, the client is referred to the server that owns the data, where the modification request can proceed.

To enable referrals only during update operations:

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

2. Under Data in the left pane, click the suffix for which you want to add a referral.

3. Click the Suffix Settings tab, and select the “Return Referrals for Update Operations” radio button.

4. Click the Referrals tab. Enter an LDAP URL in the “Enter a new referral” field, or click Construct to be guided through the creation of an LDAP URL.

   For more information about the structure of LDAP URLs, refer to Appendix C, “LDAP URLs.”

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

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

6. Click Save.

Disabling a Suffix

Sometimes, you may need to take down a database for maintenance, but the data the database contains is not replicated. Rather than returning a referral, you can disable the suffix responsible for the database.

Once you disable a suffix, 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 you want to disable.
3. Click the Suffix Setting tab, and deselect the “Enable this suffix” checkbox.

   A red dot appears on the Suffix Setting tab to alert you to changes that need to be saved.
4. Click Save.

   The suffix is no longer enabled.

Deleting a Suffix

The following procedure describes deleting a suffix:

CAUTION When you delete a suffix, you also delete 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 you want to delete.
3. Select Delete from the Object menu.

   You can also right-click the suffix and select Delete from the pop-up menu.
4. Select “Delete this suffix and all of its subsuffixes” if you want to remove all the suffix and every suffix below it.

   Select “Delete this suffix only” if you want to remove only this particular suffix, not its subsuffixes.

5. Click OK to delete the suffix.

   A progress dialog box is displayed that tells you the steps being completed by the Console.

Creating and Maintaining Databases

After you create suffixes for organizing your directory data, you create databases to contain your directory data. Databases are used to store your directory data.

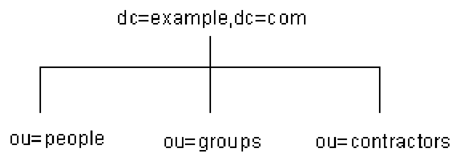
This section contains information about creating databases to contain your directory data, deleting databases, using database encryption, and making databases temporarily read-only.

Creating Databases

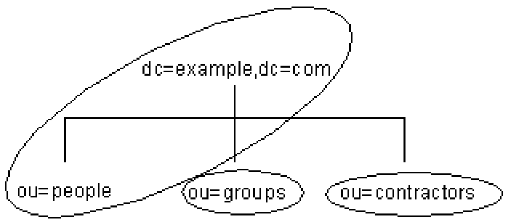
Directory Server supports the use of multiple databases over which you can distribute your directory tree. There are two ways you can distribute your data across multiple databases:

- One database per suffix.

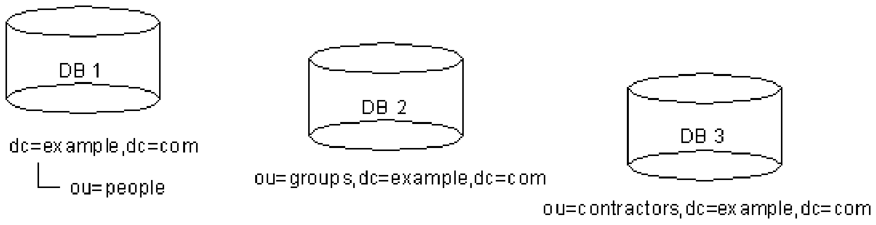
  The data for each suffix is contained in a separate database. For example, your directory tree appears as follows:



  You add three databases to store the data contained in your separate suffixes, as follows:



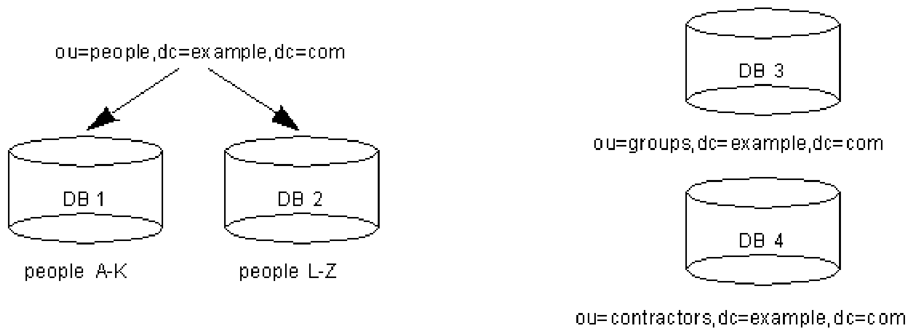
This division of the tree corresponds to three databases, as follows:



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 your directory tree is so large that you need two databases to store them. In this case, the data contained by `ou=people` could be distributed across two databases. This is illustrated as follows:


ou=people,dc=example,dc=com
DB 1
people A-K
DB 2
people L-Z
DB 3
ou=groups,dc=example,dc=com
DB 4
ou=contractors,dc=example,dc=com

Database one contains people with names from A-K, and database two contains people with names from L-Z. Database three contains the `ou=groups` data, and database four contains the `ou=contractors` data.

You need to use the custom distribution plug-in to distribute data from a single suffix across multiple databases. Contact Red Hat Professional Services for information on how to create distribution logic for your Directory Server.

Creating a New Database for an Existing Suffix Using the Console

The following procedure describes adding a database to a suffix you have already created:

1. In the Directory Server Console, select the Configuration tab.
2. In the left pane, expand Data, then click the suffix to which you want to add the new database.
3. Right-click the suffix, and select New Database from the pop-up menu.

   The "Create New Database" dialog box is displayed.
4. In the "Create New Database" dialog box, enter a unique name for the database.

   This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote ('), double quote ("), or a question mark (?). It can contain numbers; for instance, you might name the new database `example2`.

5. In the "Create database in" field, enter the path to the directory where you want to store the new database. You can also click Browse to locate a directory on your local machine.

   By default, the directory stores the new database in this directory:

   serverRoot`/slapd-`*serverID*`/db`

6. Click OK. Click Yes in the confirmation dialog to create the new database.

NOTE To see the new suffix in the Directory tab, you first need to create a root entry associated with the suffix. Refer to "Creating Directory Entries," on page 49.

Creating a New Database for a Single Suffix from the Command-Line

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

For example, you want to add a new database to the server `example1`. First, type the following to change to the directory containing the `ldapmodify` utility:

```
cd serverRoot/shared/bin
```

Add a new entry to the configuration file by performing an `ldapmodify`, as follows:

```
ldapmodify -a -h example1 -p 389 -D "cn=directory manager" -w
secret
```

The `ldapmodify` utility binds to the server and prepares it to add an entry to the configuration file.

Next, you create the entry for the new database, as follows:

```
dn: cn=UserData,cn=ldbm database,cn=plugins,cn=config
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`.

To create a root or sub suffix from the command-line, refer to "Creating Root and Sub Suffixes from the Command-Line," on page 86. The database name, given in the DN attribute, must correspond with the value in the `nsslapd-backend` attribute of the suffix entry.

Adding Multiple Databases for a Single Suffix

You can distribute a single suffix across multiple databases. However, to distribute the suffix, you need to create a custom distribution function to extend the directory. For more information on creating a custom distribution function, contact Red Hat Professional Services.

NOTE Once you have distributed entries, you cannot redistribute them. The following restrictions apply:

- You cannot change your distribution function once you have deployed entry distribution.
- You cannot use the LDAP `modrDN` operation to rename entries if that would cause them to be distributed into a different database.
- You cannot replicate distributed local databases.
- You cannot use the `ldapmodify` operation 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.

Once Red Hat Professional Services has helped you create a custom distribution logic plug-in, you need to add it to your directory. The following procedures describe adding distribution logic to a suffix in your directory.

Adding the Custom Distribution Function to a Suffix

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. You can insert a distribution function into a suffix using both the Console and the command-line.

For information about creating your own custom distribution logic, contact Red Hat Professional Services.

Adding Custom Distribution Using the Console

1. In the Directory Server Console, select the Configuration tab.
2. Expand Data in the left navigation pane. Select the suffix to which you want to apply your distribution function.

3. Select the Databases tab in the right window.
4. Click Add to associate additional databases with the suffix.

   The "Database List" dialog box is displayed. Select a database from the list, and click OK.
5. Enter the path to your distribution library in the "Distribution library" field, or click Browse to locate a distribution library on your local machine.
6. Enter the name of your distribution function in the "Function name" field.
7. Click Save to save your changes.

Adding Custom Distribution from the Command-Line

Use the `ldapmodify` command-line utility to add the following attributes to the suffix entry itself:

```
nsslapd-backend: Database1
nsslapd-backend: Database2
nsslapd-backend: Database3
nsslapd-distribution-plugin: /full/name/of/a/shared/library
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 your 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, refer to "Adding and Modifying Entries Using ldapmodify," on page 60.

Maintaining Directory Databases

This section describes jobs associated with maintaining your directory databases. It includes the following procedures:

- Placing a Database in Read-Only Mode
- Deleting a Database
- Configuring Transaction Logs for Frequent Database Updates

Placing a Database in Read-Only Mode

When a database is in read-only mode, you cannot create, modify, or delete any entries. For example, you must put a database in read-only mode if you are manually initializing a consumer.

If your Directory Server manages multiple databases, you can place all of them into read-only mode at the same time by placing your entire server in read-only mode. For more information, see "Placing the Entire Directory Server in Read-Only Mode," on page 41.

This section includes procedures for the following:

- Making a Database Read-Only Using the Console
- Making a Database Read-Only from the Command-Line

Making a Database Read-Only Using the Console

To place a database in read-only mode from the Directory Server Console:

1. In the Directory Server Console, select the Configuration tab.
2. Expand Data in the left pane. Expand the suffix containing the database you want to put in read-only mode.
3. Select the database you want to put into read-only mode.
4. Select the Database Settings tab in the right pane.
5. Select the "Database is read-only" checkbox.
6. Click Save.

Making a Database Read-Only from the Command-Line

If you want to manually place a database into read-only mode, you must change the read-only attribute, `nsslapd-readonly`, to `on`. To do so, use the `ldapmodify` command-line utility. The `nsslapd-readonly` attribute for a particular database is located in the `cn=`*database_name*`,cn=ldbm database,cn=plugins,cn=config` entry (where *database_name* is the name of the database).

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

Deleting a Database

The following procedure describes deleting a directory database using the Directory Server Console. 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. In the left navigation pane, locate the database you want to delete, and select it.
3. From the Object menu, select Delete.

   You can also right-click the database and select Delete from the pop-up menu.

   The Deleting Database confirmation dialog box is displayed.
4. Click Yes to confirm that you want to delete the database.

   A progress dialog box appears telling you the steps the Directory Server completes during the deletion.

   Once deleted, the database no longer appears in the right pane.

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.

This is done with a `dse.ldif` change. Use the `nsslapd-db-logdirectory` directive, like this:

```
nsslapd-db-logdirectory: /home3/exampledb-slapd-01-txnlogs
```

This directive goes on the same entry that has the `dbcache` size. Storing these files on a separate physical disk improves performance because the disk heads don't thrash moving between the log files and the data files.

Database 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 backend. When configured, every instance of a particular attribute, even index data, is encrypted for every entry stored in that database.

NOTE To enable database encryption on an attribute with existing stored data, you have to 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.

Indexed attributes may be encrypted, and database 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.

Encryption Keys

In order to use database encryption, the server must be configured for SSL and have SSL enabled because database 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 private 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 database 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.

CAUTION There is no mechanism for recovering a lost key. Therefore, it is especially important to backup 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.

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.

Configuring Database Encryption from the Console

1. In the Console, open the Directory Server.
2. Open the "Configuration" tab, and select the "Data" node.
3. In the "Data" node, select the backend you want to edit, such as `dc=example,dc=com`.
4. Next, select the root you want to edit, such as `o=userRoot`.
5. Select the "Attribute Encryption" tab.
6. Hit the "Add Attribute" button, and a list of attributes will appear. Select the attribute you want encrypted. A list will appear; select which encryption cipher you wish to use.

7. Repeat step 6 for every attribute you want encrypted. Then hit “Save” to save your changes.
8. To delete attributes, select them from the list of encrypted attributes in the Attribute Encryption table, and hit the “Delete” button. When you hit “Save,” a dialog box will appear asking if you want to delete the selected attributes. Click on “yes” to continue with the deletion. Any deleted attributes have to be manually re-added after you save.
9. Before you save, you can undo any changes and return to the previous configuration by pressing the “Reset” button.
10. For existing attribute entries to be encrypted, the information must be exported, then re-imported. See “Exporting and Importing an Encrypted Database,” on page 102.

Configuring Database Encryption Using the Command-Line

1. Using the `ldapmodify` command to add database encryption with the AES cipher to the `telephoneNumber` attribute, add a configuration entry like this:

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

2. For existing attribute entries to be encrypted, the information must be exported, then re-imported. See “Exporting and Importing an Encrypted Database,” on page 102.

For more information on database encryption configuration schema, refer to “Database Attributes under cn=attributeName,cn=encrypted attributes,cn=database_name,cn= ldbm database,cn=plugins,cn=config” in the *Red Hat Directory Server Configuration, Command, and File Reference*.

Exporting and Importing an Encrypted Database

Exporting and importing encrypted databases is a similar process 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 output.ldif -s "dc=example,dc=com"
-s  "o=userRoot"
```

See "Exporting to LDIF from the Command-Line," on page 161, for more information.

2. Make any configuration changes.
3. Re-import the data using the `ldif2db` script, as follows:

```
ldif2db -n Database1 -E
 -i /opt/redhat-ds/servers/slapd-dirserver/ldif/output.ldif
```

See "Importing from the Command-Line," on page 155, for more information.

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

1. 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 file named `db/guardian` and then re-start the server. This will force recovery, a side-effect of which is the deletion of 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.
2. After enabling encryption and importing data, be sure to delete the `ldif` file because it contains plaintext values for the now encrypted data. Again, steps should be taken to ensure that the disk blocks that it occupied are overwritten.
3. 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. Steps should be taken to ensure that the disk blocks that those files occupied are overwritten.
4. 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.

Additionally, 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, steps should be taken to ensure that any page or swap files are adequately protected.

Creating and Maintaining Database Links

Chaining is a method by which a server contacts other servers on behalf of a client application and then returns the combined results. This method is implemented through the database link. A database link 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.

The following sections describe how to create and configure a database link. For more general information about chaining, refer to chapter 5, "Designing the Directory Topology," in the *Red Hat Directory Server Deployment Guide*.

You can create and configure a database link using Directory Server Console or the command-line. The following sections describe the procedures for creating and maintaining a database link:

- Configuring the Chaining Policy
- Creating a New Database Link
- Chaining Using SSL
- Maintaining Database Links
- Database Links and Access Control Evaluation
- Advanced Feature: Tuning Database Link Performance
- Advanced Feature: Configuring Cascading Chaining

For information about monitoring the activity of your database links, refer to "Monitoring Database Link Activity," on page 473.

Configuring the Chaining Policy

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

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, you need to control the chaining policy so that the components can complete their operations successfully. One example is the certificate verification function. If you chain the LDAP request made by the function to check certificates, it implies that you trust the remote server. If the remote server is not trusted, then you have a security problem.

By default, all internal operations are not chained. However, you can override this default by specifying components that you want to chain using the Console or the command-line. By default, no components are allowed to chain.

You must also create an ACI on the remote server to allow the plug-in you specify to perform its operations on the remote server. You create the ACI 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 you create on the remote server:

Table 3-2 Components Allowed to Chain

Component Name	Description	Permissions
ACI Plug-in	This plug-in implements the access control feature. 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. Specify the following value in `nsActiveChainingComponents` attribute: `nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config`	Read, search, and compare
4.0 plug-ins	This component name represents all Directory Server 4.0 plug-ins. The 4.0 plug-ins share the same chaining policy. Specify the following in the `nsActiveChainingComponents` attribute: `nsActiveChainingComponents: cn=old plugin,cn=plugins,cn=config`	Depends upon the 4.0 plug-in you are allowing to chain
Resource limit component	This component sets server limits depending on the user bind DN. You can apply resource limits on remote users if the resource limitation component is allowed to chain. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=resource limits,cn=components,cn=config`	Read, search, and compare

Table 3-2 Components Allowed to Chain *(Continued)*

Component Name	Description	Permissions
Certificate-based authentication checking component	This component is used when the SASL-external bind method is used. It retrieves the user certificate from the database on the remote server. If you allow this component to chain, certificate-based authentication can work with a database link. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=certificate-based authentication,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, if you delete an entry that is a member of a group, 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, specify the following: `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 `uid` attribute are unique (no duplicates). If you allow this plug-in to chain, it confirms that the `uid` attribute values are unique even on attributes changed through a database link. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=attribute uniqueness,cn=plugins,cn=config`	Read, search, and compare

NOTE You cannot chain the following components:

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

When enabling the Referential Integrity Plug-in on servers issuing chaining requests, be sure to analyze your performance resource and time needs as well as your integrity needs. Integrity checks can be time-consuming and draining on memory/CPU.

For further information on the limitations surrounding ACIs and chaining, see "ACI Limitations," on page 208.

Once you have modified the component you allow to chain, you must restart the server in order for the modification to take effect.

The following sections describe how to specify components you want to allow to chain using the Console and from the command-line.

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. To add a component to the "Components allowed to chain" list, click Add.

   The "Select Components to Add" dialog box displays. Select a component from the list, and click OK.
4. To delete a component from the list, select it, and click Delete.

   After making modifications to the components list, a red dot appears on the tab, and the field name turns gray.
5. Click Save to save your changes.
6. Restart the server in order for the change to take effect.

After allowing the component to chain, you must create an ACI in the suffix on the remote server to which the operation will be chained. For example, you would create the following 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";)
```

Chaining Component Operations from the Command-Line

You can specify components you want 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, if you want to allow the referential integrity component to chain operations, you add the following to your database link configuration file:

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

Refer to Table 3-2, on page 105, for a list of the components you can allow to chain.

Once you have modified the `nsActiveChainingComponents` attribute, you must restart the server for your change to take effect.

After allowing the component to chain, you must create an ACI in the suffix on the remote server to which the operation will be chained. For example, you would create the following ACI for the referential integrity component:

```
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";)
```

Chaining LDAP Controls

You can choose not to 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.

- Managed DSA — This controls returns smart referrals as entries rather than following the referral. This allows you to change or delete the smart referral itself.
- Loop detection — This control keeps track of the number of times the server chains with another server. When the count reaches a number you configure, a loop is detected, and the client application is notified. For more information about using this control, refer to "Detecting Loops," on page 136.

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.

The following sections describe how to alter the controls that the database link forwards using the Console and from the command-line.

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. To add an LDAP control to the list, click Add.

   The "Select control OIDs to add" dialog box displays. Select the OID of a control you want to add to the list, and click OK.
4. To delete a control from the list, select it from the "LDAP controls forwarded to the remote server" list, and click Delete.
5. After making modifications to the components list, a red dot appears on the tab, and the components field name turns gray. Click Save to save your changes.

Chaining LDAP Controls from the Command-Line

You can 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, you add the following to your database link entry in the configuration file:

```
nsTransmittedControls: 2.16.840.1.113730.3.4.9
```

In addition, if clients of your Directory Server create their own controls and you want their operations to be chained to remote servers, you need to add the OID of the custom control to the `nsTransmittedControls` attribute.

The LDAP controls you can chain and their OIDs are listed in the following table:

Table 3-3 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

For more information about LDAP controls, refer to the LDAP C-SDK documentation at `http://www.redhat.com/docs/manuals/dir-server/`.

Creating a New Database Link

The basic configuration of your database link involves providing the following information:

- **Suffix information.** You create a suffix in your 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. You need to specify the DN and the credentials you want each database link to use to bind with remote servers.
- **LDAP URL.** You provide the LDAP URL of the remote server to which the database link connects.
- **List of failover servers.** You can provide a list of alternative servers for the database link to contact in the event of a failure. This configuration item is optional.

The following sections describe creating a new database link from the Directory Server Console as well as the command-line.

Creating a New Database Link Using the Console

To create a new database link using the Directory Server 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 or New Sub Suffix from the pop-up menu.

   A "Create New Suffix" dialog box is displayed.
3. Enter the name of the suffix on the remote server to which you want to chain in the "New suffix" field.

   The suffix must be named according to dc naming conventions. For example, you might enter a new suffix name of `dc=example,dc=com`.
4. Deselect the "Create associated database automatically" checkbox.

   You deselect the checkbox because you cannot add a database link to a suffix that is associated with a database. This suffix is used only by the database link.
5. Click OK to create the new suffix.

   The suffix appears automatically under the Data branch in the left navigation pane.
6. In the left pane, right-click the suffix you just created, and select "New Database Link" from the pop-up menu.

   The Create New Database Link dialog box is displayed.
7. Enter the name of the new database link in the "Database link name" field.

   Use only ASCII (7-bit) characters for naming the database link. This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote ('), double quote ("), or a question mark (?). For example, you might name the new database link `examplelink1`.
8. Enter the DN used by the database link to bind to the remote server in the "Bind DN" field.

   For example, you might enter `cn=dblink` in the "Bind DN" field.
9. Enter the password used by the database link to bind to the remote server in the "Password" field.
10. Select the "Use a secure LDAP connection between servers" checkbox if you want the database link to use SSL to communicate to the remote server.

11. Enter the name of the remote server in the "Remote server" field. Enter the server port number used for the bind in the "Remote server port" field. The default port number is `389`.

12. Enter the name of a failover server in the "Failover Server(s)" field, and specify a port number in the "Port" field. The default port number is `389`. Click Add to add the failover server to the list.

    You can specify multiple failover servers. If the primary remote server fails, the database link contacts the first server in the Failover Servers list. If it fails, it contacts the next in the list, and so on.

13. Click OK to create the new database link. Click OK to dismiss the success dialog box that appears after the database link has been created.

    Your new database link appears under the suffix in the left navigation pane.

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

Creating a Database Link from the Command-Line

Use the `ldapmodify` command-line utility to create a new database link from the command-line.

Your new instance must be located in the `cn=chaining database,cn=plugins, cn=config` entry.

Default configuration attributes are contained in the `cn=default 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. You cannot change the default configuration attributes on existing database links.

Each database link contains its own specific configuration information, which is stored with the database link entry itself, `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config`. For more information about configuration attributes, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

This section contains the following procedures for configuring a database link from the command-line:

- Providing Suffix Information
- Providing Bind Credentials
- Providing an LDAP URL
- Providing a List of Failover Servers
- Summary of Cascading Chaining Configuration Attributes
- Database Link Configuration Example

Providing Suffix Information

Use the `nsslapd-suffix` attribute to define the suffix managed by your database link. For example, if you want your database link to point to the people information for a remote site of your company, you would 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_name*`,cn=chaining database,cn=plugins,cn=config` entry.

NOTE	After creation time, any alterations you make to the `nsslapd-suffix` attribute occur only after you have restarted the server containing the database link.

Providing Bind Credentials

For a request from a client application to be chained to a remote server, you can provide special bind credentials for the client application. This gives the remote server the proxied authorization rights needed to chain operations. If you do not specify bind credentials, the database link binds to the remote server as anonymous.

Providing bind credentials involves the following steps:

1. On the remote server, you need to do the following:

    a. Create an administrative user for the database link.

       For information on adding entries, see “Creating Directory Entries,” on page 47.

b. 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, refer to "Managing Access Control," on page 205.

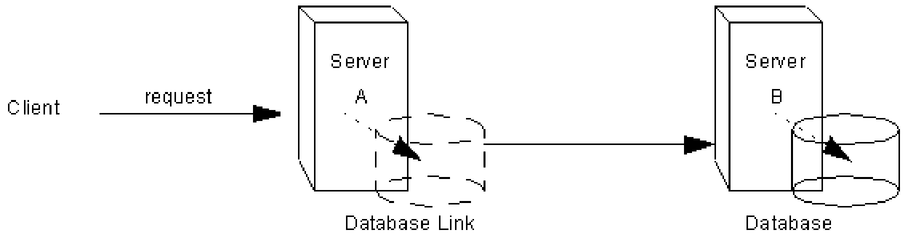
2. On the server containing the database like, you need to do the following:

   a. Use `ldapmodify` to provide a user DN for the database link in the `nsMultiplexorBindDN` attribute of the `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config` entry.

CAUTION The `nsMultiplexorBindDN` cannot be that of the Directory Manager.

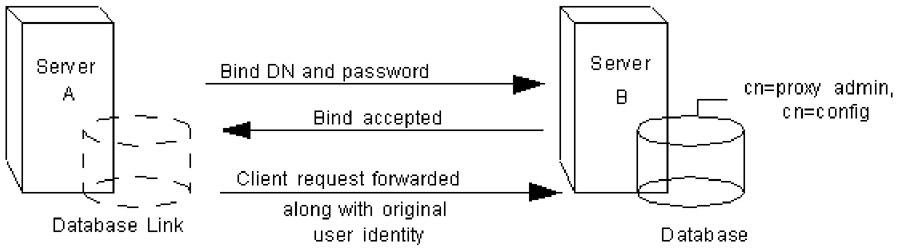
   b. Use `ldapmodify` to provide a user password for the database link in the `nsMultiplexorCredentials` attribute of the `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config` entry.

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 A
Database Link
Bind DN and password
Bind accepted
Client request forwarded along with original user identity
Server B
cn=proxy admin, cn=config
Database

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

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

For more information on ACIs, refer to "Managing Access Control," on page 205. For more information about the proxy authentication control, refer to the C-SDK documentation at `http://www.redhat.com/docs/manuals/dir-server/`.

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.

Providing an LDAP URL

On the server containing your database link, you have to 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 following form:

```
ldap://hostname:portnumber/
```

You specify the URL of the remote server using the `nsFarmServerURL` attribute in the `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config` entry of the configuration file. For example, the `nsFarmServerURL` might appear as follows:

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

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

If you want the database link to connect to the remote server using LDAP over SSL, the LDAP URL of the remote server takes the following form:

```
ldaps://hostname:portnumber/
```

For more information about chaining and SSL, refer to "Chaining Using SSL," on page 121.

Providing a List of Failover Servers

You can include additional LDAP URLs for servers to use in the case of failure. To do so, add alternate servers to the `nsFarmServerURL` attribute, separated by spaces. For example, you might enter the following:

```
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`.

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.

Attributes marked with an asterisk (*) can be both global or instance attributes. All instance attributes are defined in the `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config` entry.

The two global configuration attributes are located in the `cn=config,cn=chaining database,cn=plugins,cn=config` entry. The global attributes are dynamic, meaning any changes you make to them will automatically take effect on all instances of the database link within your directory.

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

Table 3-4 Database Link Configuration Attributes

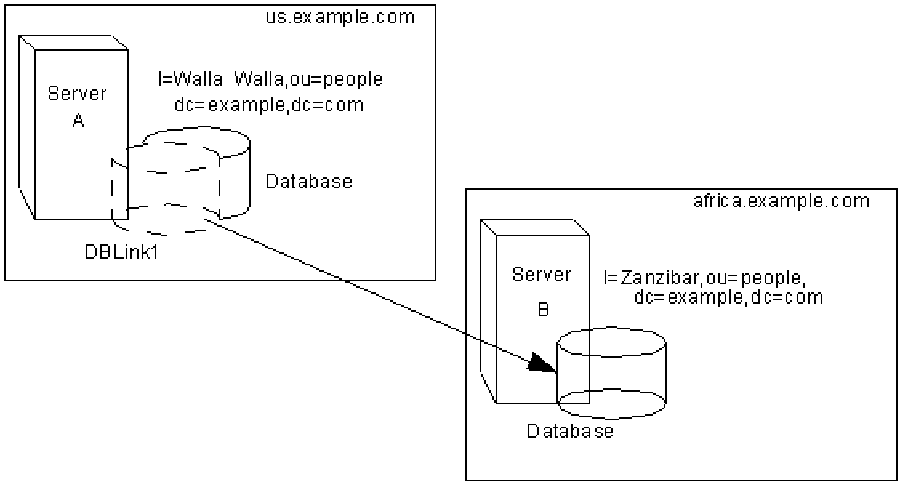
Attributes	Value
`*nsTransmittedControls`	Gives the OID of LDAP controls forwarded by the database link to the remote data server.
`nsslapd-suffix`	The suffix managed by the database link. Any changes you make to this attribute after the entry has been created take effect only after you restart the server containing the database link.
`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.
`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. Allows you to disable proxied authorization. A value of `off` means proxied authorization is disabled. The default value is `on`.

Table 3-4 Database Link Configuration Attributes *(Continued)*

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. Refer to "Chaining Component Operations," on page 104, for more information.
`nsReferralOnScopedSearch`	Controls whether referrals are returned by scoped searches. This attribute is for optimizing your 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`.

Database Link Configuration Example

Suppose you have a server within the `us.example.com` domain that contains the subtree `l=Walla Walla,ou=people,dc=example,dc=com` on a database and that you want to chain operation requests for `l=Zanzibar,ou=people,dc=example,dc=com` to a different server in the `africa.example.com` domain. This operation is illustrated in the following diagram:



First, use the `ldapmodify` command-line utility to add a database link to server A. Type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Run the script, as follows:

```
ldapmodify -a -p 389 -D "cn=directory manager" -w secret -h
 us.example.com
```

Then specify the configuration information for the database link:

```
dn: cn=DBLink1,cn=chaining database,cn=plugins,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: l=Zanzibar,ou=people,dc=example,dc=com
nsfarmserverurl: ldap://africa.example.com:389/
nsmultiplexorbinddn: cn=proxy admin,cn=config
nsmultiplexorcredentials: secret
cn: DBLink1

dn: cn="l=Zanzibar,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: l=Zanzibar,ou=people,dc=example,dc=com
```

In the first section, the `nsslapd-suffix` attribute contains the suffix on server B to which you want to chain from server A. The `nsFarmServerURL` attribute contains the LDAP URL of server B.

The second section 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 `nssalpd-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`.

Next, you 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
```

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

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, you need to 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, refer to "Database Links and Access Control Evaluation," on page 123.

Chaining Using SSL

You can configure your database links to communicate with the remote server using SSL. Using SSL to chain involves the following steps:

- Enable SSL on the remote server.

  For more information on enabling SSL, refer to "Enabling SSL: Summary of Steps," on page 426.

- Specify the LDAP URL of the remote server in SSL format.

  You specify the LDAP URL in the `nsFarmServerURL` attribute. For more information about this attribute, see "Providing an LDAP URL," on page 116.

  For example, you might specify the following LDAP URL:

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

- Enable SSL on the server that contains the database link.

  For more information on enabling SSL, refer to "Enabling SSL: Summary of Steps," on page 426.

When you configure the database link and remote server 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.

Maintaining Database Links

This section describe how to update and delete existing database links. It contains the following procedures:

- Updating Remote Server Authentication Information
- Deleting Database Links

Updating Remote Server Authentication Information

To update the bind DN and password used by the database link to connect to the remote server:

1. In the Directory Server Console, select the Configuration tab.
2. In the left pane, expand Data, and locate the database link you want to update under one of the suffixes. Select the database link.
3. In the right navigation pane, click the Authentication tab.
4. To update the remote server information, enter a new LDAP URL in the "Remote Server URL" field.

   Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It takes the following form:

   `ldap://`*servername*`:`*portnumber*`/`
5. Update the bind DN used by the database link to bind with the remote server by entering a new DN in the "Database link bind DN" field.
6. Update the password used by the database link to bind with the remote server by entering a new password in the "Database link password" field. Confirm the password by retyping it in the "Confirm database link password" field.

   The remote server checklist box lists the administrative user entry, suffix, and ACI that need to exist on the remote server for the database link to bind successfully.
7. Click Save to save your changes.

Deleting Database Links

To delete a database link:

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

2. In the left navigation pane, locate the database link you want to delete, and select it.

3. From the Object menu, select Delete.

You can also right-click the database link and select Delete from the pop-up menu.

The Deleting Database Link confirmation dialog box is displayed.

4. Click Yes to confirm that you want to delete the database link.

A progress dialog box appears telling you the steps the Directory Server completes during the deletion.

Once deleted, the database link no longer appears in the right pane.

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 via 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 means that you need to add the usual access controls to the remote server with a few restrictions:

- You cannot use all types of access control.

  For example, role-based or filter-based ACIs need access to the user entry. Because you are accessing the data via database links, only the data in the proxy control can be verified. Consider designing your 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.

The following restrictions apply to the ACIs you create to use 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 may refer to remote users.

- ACIs must be located with any role definitions they use and with any users intended to have those roles.
- ACIs that refer 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, you can also choose to have them 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_name*`,cn=chaining database,cn=plugins,cn=config` entry. However, evaluating access controls on the server containing the database link is not recommended unless using cascading chaining.

Advanced Feature: Tuning Database Link Performance

The following sections provide information on tuning the performance of your database links through connection and thread management. It contains the following parts:

- Managing Connections to the Remote Server
- Detecting Errors During Normal Processing

- Managing Threaded Operations

Managing Connections to the Remote Server

Each database link maintains a pool of connections to a remote server. You can configure the connections to optimize resources for your directory.

You can change the connection attributes using the Directory Server Console or through the command-line.

Managing Connections to the Remote Server Using the Console

1. In the Directory Server Console, select the Configuration tab.
2. Expand the Data folder in the left pane and locate the database link you want to change. Click the database link, then click the Limits and Controls tab in the right navigation pane.
3. In the Connection Management section, make changes to any of the following fields:
    - **Maximum TCP connection(s).** The maximum number of TCP connections that the database link establishes with the remote server. The default value is `3` connections.
    - **Bind timeout.** Amount of time, in seconds, before the database link's bind attempt times out. The default value is `15` seconds.
    - **Maximum binds per connection.** Maximum number of outstanding bind operations per TCP connection. The default value is `10` outstanding bind operations per connection.
    - **Time out before abandon (sec).** Number of seconds before the server checks to see if a timed-out connection should be abandoned. The default value is `2` seconds.
    - **Maximum LDAP connection(s).** Maximum number of LDAP connections that the database link establishes with the remote server. The default value is `10` connections.
    - **Maximum bind retries.** Number of times a database link attempts to bind to the remote server. A value of `0` indicates that the database link will try to bind only once. The default value is `3` attempts.
    - **Maximum operations per connection.** Maximum number of outstanding operations per LDAP connection. The default value is `10` operations per connection.

- **Connection lifetime (sec).** How long a connection made between the database link and remote server remains open. You can keep connections between the database link and the remote server open for an unspecified time, or you can close them after a specific period of time.

  It is faster to keep the connections open, but it uses more resources. For example, if you are using a dial-up connection, you may want to limit the connection time.

  A value of `0` indicates there is no limit. By default, the value is set to `0`.

4. Click Save to save your changes.

Managing Connections to the Remote Server from the Command-Line

Use `ldapmodify` to add connection attributes to your database link entry.

The default connection management attributes are stored in the following entry:

```
cn=default instance config, cn=chaining
database,cn=plugins,cn=config
```

The connection management attributes for a specific database link are stored in the following entry:

```
cn=database_link_name,cn=chaining database,cn=plugins,cn=config
```

where *database_link_name* is the name of the database link. The connection management attributes specified in this entry take precedence over the attributes specified in the `cn=default instance config` entry.

The following table lists the attributes associated with connection management:

Table 3-5 Database Link Connection Management Attributes

Attribute Name	Description
`nsOperationConnectionsLimit`	Maximum number of LDAP connections that the database link establishes with the remote server. The default value is `10` connections per database link instance.
`nsBindConnectionsLimit`	Maximum number of TCP connections that the database link establishes with the remote server. The default value is `3` connections.
`nsConcurrentOperationsLimit`	Maximum number of outstanding operations per LDAP connection. The default value is `10` operations per connection.
`nsConcurrentBindLimit`	Maximum number of outstanding bind operations per TCP connection. The default value is `10` outstanding bind operations.

Table 3-5 Database Link Connection Management Attributes *(Continued)*

Attribute Name	**Description**
`nsBindRetryLimit`	Number of times a database link attempts to bind to the remote server. A value of zero (`0`) indicates that the database link will try to bind only once. The default value is `3` attempts.
`nsConnectionLife`	Connection lifetime, in seconds. You can keep connections between the database link and the remote server open for an unspecified time, or you can close them after a specific period of time. It is faster to keep the connections open, but it uses more resources. For example, if you are using a dial-up connection, you may want to limit the connection time. A value of `0` indicates there is no limit. By default, the value is set to `0`. When the value is `0` and you provide a list of failover servers in the `nsFarmServerURL` attribute, the "main" server is never contacted after failover to the alternate server. The default value is `0` seconds.
`nsBindTimeout`	Amount of time, in seconds, before the bind attempt times out. The default value is `15` seconds.
`nsAbandonedSearchCheckInterval`	Number of seconds that pass before the server checks for abandoned operations. The default value is `2` seconds.

For the list of database link configuration attributes, refer to "Database Link Configuration Attributes," on page 117.

Detecting Errors During Normal Processing

You can help protect server performance by detecting errors during the normal chaining operation between the database link and the remote server. The database link has two attributes which work together to determine if the remote server is no longer responding.

The first attribute, `nsMaxResponseDelay`, sets a maximum duration for an LDAP operation to complete. If the operation takes more than the amount of time specified in this attribute, the database link's server suspects that the remote server is no longer online.

Once the `nsMaxResponseDelay` period has been met, the database link pings the remote server. During the ping, the database link issues another LDAP request, a simple search request for an object that does not exist in the remote server. The duration of the ping is set using the `nsMaxTestResponseDelay`.

If the remote server does not respond before the `nsMaxTestResponseDelay` period has passed, then an error is returned, and the connection is flagged as down. All connections between the database link and remote server will be blocked for 30 seconds, protecting your server from a performance degradation. After 30 seconds, operation requests made by the database link to the remote server continue as normal.

Both attributes are stored in the `cn=config,cn=chaining database,cn=plugins,cn=config` entry. The following table describes the attributes in more detail:

Table 3-6 Database Link Processing Error Detection Parameters

Attribute Name	**Description**
`nsMaxResponseDelay`	Maximum amount of time it can take a remote server to respond to an LDAP operation request made by a database link before an error is suspected.This period is given in seconds. The default delay period is `60` seconds. Once this delay period has been met, the database link tests the connection with the remote server.
`nsMaxTestResponseDelay`	Duration of the test issued by the database link to check whether the remote server is responding. If a response from the remote server is not returned before this period has passed, the database link assumes the remote server is down, and the connection is not used for subsequent operations. This period is given in seconds. The default test response delay period is `15` seconds.

Managing Threaded Operations

Generally, Directory Server performs best using a limited number of threads for processing operations. A limited number of threads can generally process operations very quickly, preventing the queue of operations waiting for a free thread from growing too long.

However, the database link forwards operations to remote servers for processing. The database link contacts the remote server, forwards the operation, waits for the result, and then sends the result back to the client application. The entire operation can take much longer than a local operation.

While the database link waits for results from the remote server, it can process additional operations. By default, the number of threads used by the server is `20`. However, when using database links, you can improve performance by increasing the number of threads available for processing operations. While the local CPU waits for a response from a remote server, it can process other operations rather than stand idle.

To change the number of threads used for processing operations, change the `nsslapd-threadnumber` global configuration attribute in the `cn=config` entry. The default thread number is `20`. For example, you can increase the thread number to `50` to improve performance. After changing the thread number, restart the server to implement your changes.

Advanced Feature: Configuring Cascading Chaining

You can configure your database link 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.

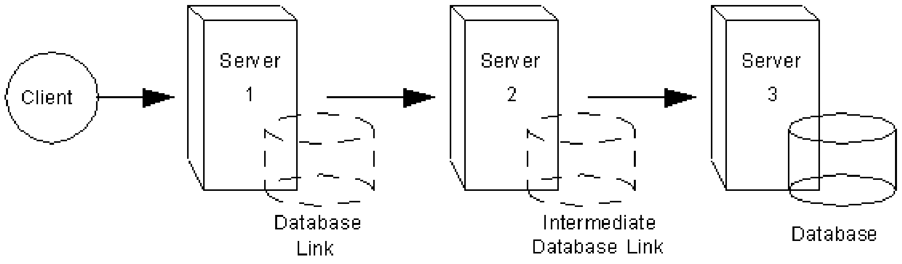
The section contains the following topics:

- Overview of Cascading Chaining
- Configuring Cascading Chaining Defaults Using the Console
- Configuring Cascading Chaining Using the Console
- Configuring Cascading Chaining from the Command-Line
- Summary of Cascading Chaining Configuration Attributes
- Cascading Chaining Configuration Example

Overview of Cascading Chaining

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

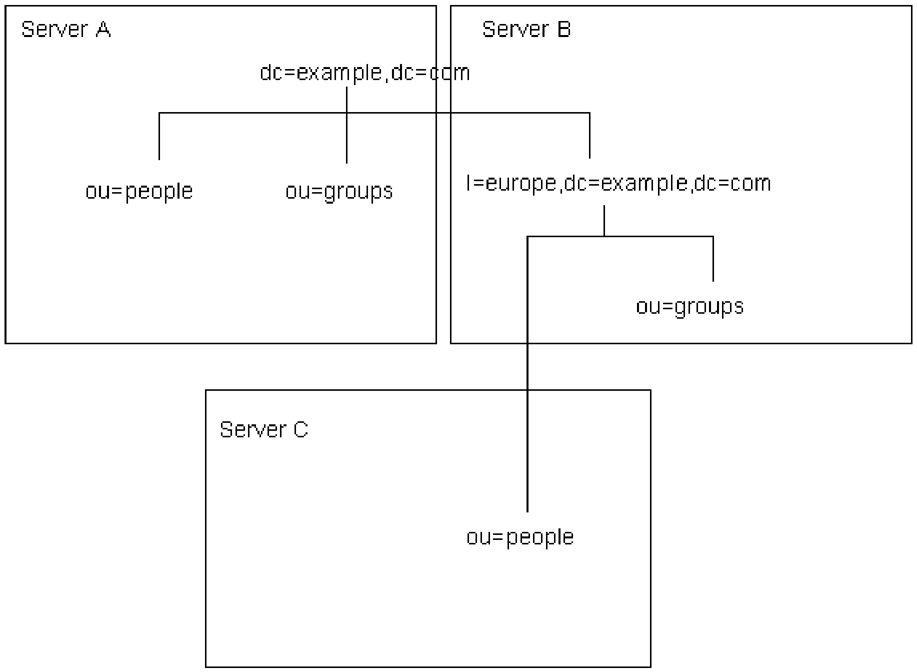
For example, consider the following scenario:



The client application sends a modify request to server one. Server one contains a database link that forwards the operation to server two, which contains another database link. The database link on server two 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.

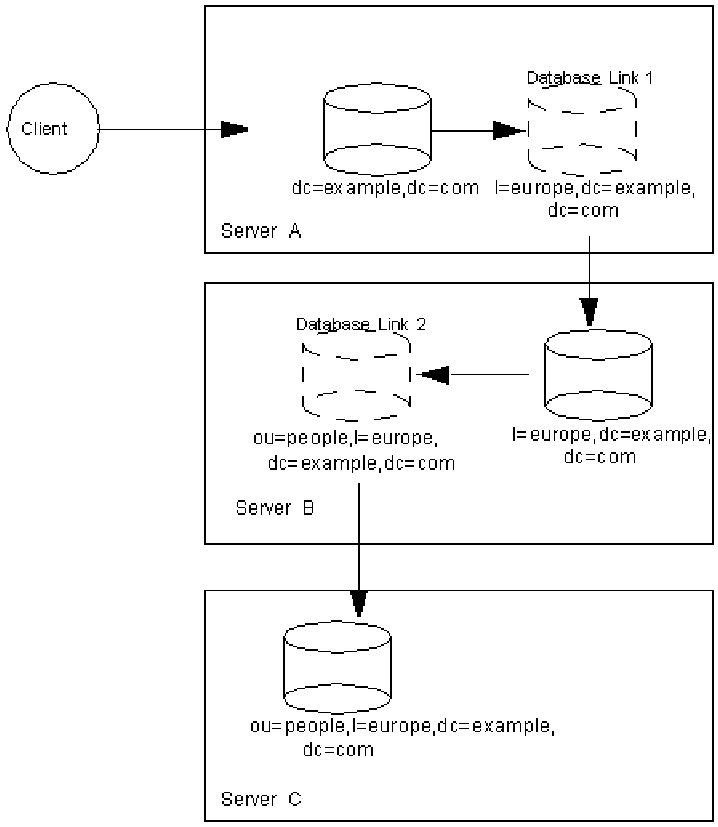
During a normal operation request, a client binds to the server, and then any ACIs applying to that client are evaluated. With cascading chaining, the client bind request is evaluated on server one, but the ACIs applying to the client are evaluated only after the request has been chained to the destination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

Consider the following example scenario. On server A, a directory tree is split as follows:



The root suffix `dc=example,dc=com` and the `ou=people` and `ou=groups` subsuffixes 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.

Configuring Cascading Chaining Defaults Using the Console

To set cascading chaining defaults for all database links in your Directory Server:

1. In the Directory Server Console, select the Configuration tab.
2. Expand the Data folder in the left pane, and click Database Link Settings. Click the Default Creation Parameters tab.

3. Select the "Check local ACI" checkbox if you want to enable the evaluation of local ACIs on the intermediate database links involved in cascading chaining. If you select this checkbox, you will need to add the appropriate local ACIs to a database on the servers that contain intermediate database links.

   This is an advanced feature. For more information, refer to "Enabling Local ACI Evaluation," on page 135.

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 `10` hops. After `10` hops, a loop is detected by the server, and an error is returned to the client application.

5. Click Save to save your changes.

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

Configuring Cascading Chaining Using the Console

To configure cascading chaining for a particular set of database links, do the following:

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

2. Expand the Data folder in the left pane, and locate the database link you want to include in a cascading chain. Click the database link, then click the Limits and Controls tab in the right navigation pane.

3. Select the "Check local ACI" checkbox if you want to enable the evaluation of local ACIs on the intermediate database links involved in the cascading chain. If you select this checkbox, you may need to add the appropriate local ACIs to the database link.

   This is an advanced feature. For more information, refer to "Enabling Local ACI Evaluation," on page 135.

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 10 hops. After 10 hops, a loop is detected by the server, and an error is returned to the client application.

5. Click Save to save your changes.

Configuring Cascading Chaining from the Command-Line

Configuring a cascade of database links through the command-line involves the following steps:

- Pointing one database link to the URL of the server containing the intermediate database link.
- Configuring the intermediate database link or links (in the example, server two) to transmit the Proxy Authorization Control.
- Creating a proxy administrative user ACI on all intermediate database links. To do so, you will need to create a database on each server that contains an intermediate database link.
- Enabling local ACI evaluation on all intermediate database links.
- Creating client ACIs on all intermediate database links and the final destination database.

Pointing to Another 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`. The `cn=`*database_link_name*`,cn=chaining database, cn=plugins,cn=config` entry of the database link on server one would contain the following:

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

Transmitting 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, refer to “Chaining LDAP Controls,” on page 108.

Creating the Proxy Administrative User ACI

You need to create an ACI 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 two does not check the credentials of server one, then anyone could bind as `anonymous` and pass a proxy authorization control allowing them more administrative privileges than appropriate.

To prevent this security hole, you need to create an ACI on the server which contains the intermediate database link. To create an ACI, you need to do the following:

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 "Creating Databases," on page 92.
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. Add the following ACI to the administrative user's entry:

   ```
   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 you create on the remote server when configuring simple chaining.

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

Enabling Local ACI Evaluation

To confirm that the proxy administrative ACI is used, you need to enable evaluation of local ACIs on all intermediate database links involved in chaining. To do this, add the following attribute to the `cn=`*database_link_name*`,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_name*`,cn=chaining database,cn=plugins,cn=config` entry.

Creating Client ACIs

Because you have enabled local ACI evaluation, you need to create the appropriate client application ACIs on all intermediate database links, as well as the final destination database.

To do this on the intermediate database links, you first need to create a database that contains a suffix that represents a root suffix of the final destination suffix.

For example, if you are chaining a client request made to the `c=africa,ou=people,dc=example,dc=com` suffix on a remote server, all intermediate database links need to contain a database associated with the `dc=example,dc=com` suffix.

You then need to add any client ACIs to this superior suffix entry. For example, you might add the following

```
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 one to perform any type of operation on the data below the `ou=people,dc=example,dc=com` suffix on server three.

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.

Summary of Cascading Chaining Configuration Attributes

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

Table 3-7 Cascading Chaining Configuration Attributes

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

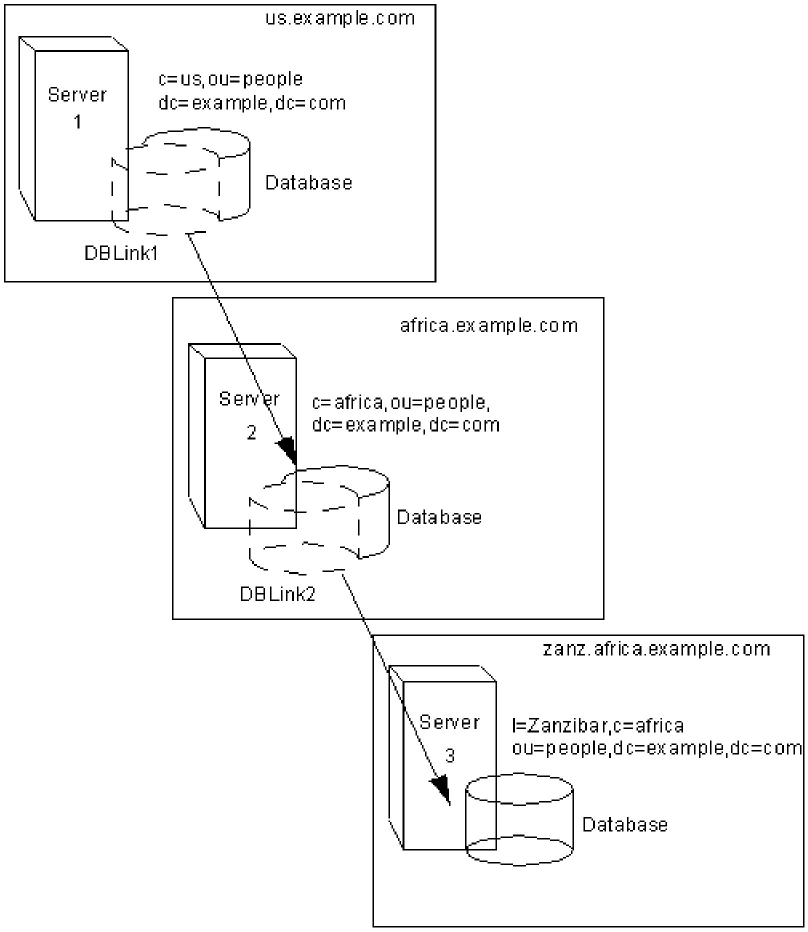
`nsTransmittedControls`	Enter the following OIDs to the database links involved in the cascading chain:
	`nsTransmittedControls: 2.16.840.1.113730.3.4.12`
	`nsTransmittedControls: 1.3.6.1.4.1.1466.29539.12`
	The first OID corresponds to the Proxy Authorization Control. The second OID corresponds to the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

Cascading Chaining Configuration Example

To create a cascading chain involving three servers as in the diagram below, you have to configure the chaining components on all three servers. This section describes the configuration steps for creating a cascading chain involving three servers and is divided into the following sections:

- Configuring Server One
- Configuring Server Two
- Configuring Server Three



Configuring Server One

First, use the `ldapmodify` command-line utility to add a database link to server one. To use the utility, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Run the utility, as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

Then specify the configuration information for the database link, DBLink1, on server one, as follows:

```
dn: cn=DBLink1,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://africa.example.com:389/
nsmultiplexorbinddn: cn=server1 proxy admin,cn=config
nsmultiplexorcredentials: secret
cn: DBLink1
nsCheckLocalACI:off

cn="l=Zanzibar,c=africa,ou=people,dc=example,dc=com",cn=mapping
tree,cn=config
objectclass=nsMappingTree
nsslapd-state=backend
nsslapd-backend=DBLink1
nsslapd-suffix: l=Zanzibar,c=africa,ou=people,dc=example,dc=com
cn: l=Zanzibar,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. You do not need to configure the `nsCheckLocalACI` attribute to check local ACIs, as this is only required on the database link, DBLink2, on server two.

Since you want to implement loop detection, you need 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 one. You specify the OID, as follows:

```
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, you do not need to perform this configuration step.

Configuring Server Two

Next, you create a proxy administrative user on server two. This administrative user will be used to allow server one to bind and authenticate to server two. It is useful to choose a proxy administrative user name which is specific to server one, as it is the proxy administrative user which will allow server *one* to bind to server two. You 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
```

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

Next, you configure the database link, DBLink2, on server two. Using `ldapmodify`, specify the configuration information for DBLink2, as follows:

```
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 your cascading chaining configuration, you need to set the `nsCheckLocalACI` to `on` to allow the server to check whether it should allow the client and proxy administrative user access to the database link.

The database link on server two 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, you need to specify both corresponding OIDs. Add the following information to the `cn=config,cn=chaining database, cn=plugins,cn=config` entry on server two:

```
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
```

where `nsTransmittedControl: 2.16.840.1.113730.3.4.12` is the OID for the proxy authorization control and `nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12` is the OID for the loop detection control.

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

The next step is to configure your ACIs. On server two, you need to ensure that a suffix exists above the `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` suffix to allow you to:

- add the database link suffix,
- add a local proxy authorization ACI that will be used to allow server one to connect using the proxy authorization administrative user that will be created on server two, and
- add a local client ACI that allows the client operation to succeed on server two, so that it can be forwarded to server three. This local ACI is needed because you have turned local ACI checking 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, it is assumed that the database corresponding to the `c=africa,ou=people,dc=example,dc=com` suffix already exists 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.

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";)
```

Then add the local client ACI that will allow the client operation to succeed on server two, given that ACI checking is turned on. This ACI is the same as the ACI you will create on the destination server to provide access to the `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` branch. You may decide that you want all users within `c=us,ou=people,dc=example,dc=com` to have update access to the entries in `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` on server three. The following ACI is the ACI you would need to create on the `c=africa,ou=people,dc=example,dc=com` suffix on server two 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 one to perform any type of operation on the `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` suffix tree on server three. Should you have users on server two under a different suffix that will require additional rights on server three, you may need to add additional client ACIs on server two.

Configuring Server Three

The final configuration step in our cascading chaining example is to configure server three. First, you create an administrative user on server three for server two 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
```

Then you need to add the same local proxy authorization ACI to server three as you did on server two. 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 two 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.

You then need to 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 you created for the client on server two:

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

Once you have completed all these steps, your cascading chaining configuration is set up. This cascading configuration will allow you to bind to server one and modify information in the `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` branch on server three. Depending on your security needs, you may want to provide more detailed access control.

Using Referrals

You can use referrals to 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:

- Setting Default Referrals
- Creating Smart Referrals
- Creating Suffix Referrals

For conceptual information on how you can use referrals in your directory, see *Red Hat Directory Server Deployment Guide*.

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 your directory. The following procedures describes setting a default referral for your directory using the Console and the command-line utilities.

Setting a Default Referral Using the Console

Set a default referral to your directory, as follows:

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 in the “Referrals to” text box, and click OK.

   For example:

   ```
   ldap://directory.example.com:389/dc=example,dc=com
   ```

   You can enter multiple referral URLs separated by spaces and in quotes, as follows:

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

For more information about LDAP URLs, refer to Appendix C, “LDAP URLs.”

Setting a Default Referral from the Command-Line

Use the `ldapmodify` command-line utility to add a default referral to the `cn=config` entry in your directory's configuration file.

For example, to add a new default referral from your Directory Server, `dir1.example.com`, to a server named `dir2.example.com`, add a new line to the `cn=config` entry. First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Then, run the `ldapmodify` utility, as follows:

```
ldapmodify -h dir1.example.com -p 389 -D "cn=directory manager"
-w secret
```

The `ldapmodify` utility binds to the server and prepares it to change an entry in the configuration file.

Next, you add the default referral to the `dir2.example.com` server:

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

Once you have added the default referral to the `cn=config` entry of your directory, the directory will return the default referral in response to requests made by client applications. You do not need to restart the server.

Creating Smart Referrals

Smart referrals allow you to map a directory entry or directory tree to a specific LDAP URL. Using smart referrals, you can refer client applications 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`. You return a smart referral to the client which 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. For more information, go to `http://www.ietf.org/rfc/rfc2251.txt` to read the RFC.

The following procedures describe creating smart referrals using both the Console and the command-line utilities.

Creating Smart Referrals Using the Directory Server Console

To configure *smart* referrals:

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 you want to add the referral.
3. Right-click the entry, and select Set Smart Referrals.

   The Edit Smart Referrals dialog box shows up.
4. Select the "Enable Smart Referral" option to define smart referrals for the selected entry. (Unchecking the option removes all smart referrals from the entry and deletes `objectclass: referral`.)
5. In the "Enter a new Smart Referral" field, enter a referral in the LDAP URL format, and then click Add to add the referral to the list. The LDAP URL to which you want to refer client application requests must be in the following format:

   ```
   ldap://hostname:portnumber/[optional_dn]
   ```

   where `[`*optional_dn*`]` is the explicit DN you want the server to return to the requesting client application. For example, you might enter an LDAP URL, as follows:

   ```
   ldap://directory.example.com:389/cn=john
   doe,o=people,l=europe,dc=example,dc=com
   ```

   If you want the server to use the DN from the original search request instead, enter the LDAP URL in the format:

   ```
   ldap://hostname:portnumber/
   ```

   You may also click Construct to be guided through the process of adding a referral.

   To allow a referral to be followed with different authentication, click Authentication, and specify the appropriate DN and password. Keep in mind that this authentication remains valid only until the Console is closed; then, it's reset to the same authentication used to log into the Console.

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 when you select “Return Referrals for All Operations” or “Return Referrals for Update Operations” 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. Click OK to save your changes.

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 is expected to 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, you must use `%20` instead of spaces in any LDAP URL you intend to use as a referral.

To add the entry `uid=jdoe,ou=people,dc=example,dc=com` with a referral to `directory.europe.example.com`, you would include the following in your 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 information about the `ldapmodify` utility, see *Red Hat Directory Server Configuration, Command, and File Reference*. For more information on smart referrals, see *Red Hat Directory Server Deployment Guide*.

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. For more information about referrals, refer to *Red Hat Directory Server Deployment Guide*.

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

Creating Suffix Referrals Using the Console

To create a suffix referral using the Console:

1. In the Directory Server Console, select the Configuration tab.
2. Under Data in the left pane, click the suffix to which you want to add a referral.
3. In the Suffix Settings tab, select one of the following radio buttons:
   - **Return Referrals for all Operations.** This means that a referral will be returned when this suffix receives any request from a client application.
   - **Return Referrals for Update Operations.** This means a referral will be returned when this suffix receives an update request from a client application. This option is used to redirect update and write requests made by client applications to a read-only database.
4. Click the Referrals tab. Enter an LDAP URL in the “Enter a new referral” field, or click Construct to be guided through the creation of an LDAP URL.

   For more information about the structure of LDAP URLs, refer to Appendix C, “LDAP URLs.”

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

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

6. Click Save.

Creating Suffix Referrals from the Command-Line

Use the `ldapmodify` command-line utility to add a suffix referral to an entry in your directory configuration file. The suffix referral information is added to the root or sub suffix entry under the `cn=mapping tree,cn=config` branch.

For example, to add a new suffix referral to the `ou=people,dc=example,dc=com` root suffix, you run `ldapmodify`. First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Then, run `ldapmodify`, as follows:

```
ldapmodify -a -h example.com -p 389 -D "cn=directory manager" -w
secret
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add a suffix referral to the `ou=people,dc=example,dc=com` root suffix, as follows:

```
dn: cn="ou=people,dc=example,dc=com",cn=mapping tree,cn=config
objectclass: extensibleObject
objectclasss: 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.

You can also set the `nsslapd-state` attribute 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, refer to Table 3-1, on page 87.

Chapter 4

Populating Directory Databases

Databases contain the directory data managed by your Red Hat Directory Server (Directory Server). This chapter describes the following procedures for populating your directory databases:

- Importing Data (page 151)
- Exporting Data (page 158)
- Backing Up and Restoring Data (page 161)
- Enabling and Disabling Read-Only Mode (page 168)

Importing Data

Directory Server provides three methods for importing data:

- Import from the Directory Server Console — You can use the Directory Server Console to append data to all of your databases, including database links.
- Initialize databases — You can use the Directory Server Console to import data to one database. This method overwrites any data contained by the database.
- Importing data from the command-line — You can import data using the command-line utilities.

NOTE The LDIF files you use for import operations must use UTF-8 characterset encoding. Import operations do not convert data from local characterset encoding to UTF-8 characterset encoding.

Table 4-1 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 speciality	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

The following sections describe importing data:

- Importing a Database from the Console
- Initializing a Database from the Console
- Importing from the Command-Line

CAUTION All imported LDIF files must also contain the root suffix.

Importing a Database from the Console

When you perform 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 your Directory Server and on remote databases to which your Directory Server has a configured database link.

You must be logged in as the Directory Manager in order to perform an import.

To import data from the Directory Server Console:

1. In the Directory Server Console, select the Tasks tab. Scroll to the bottom of the screen, and select Import Database.

   You can also import by going to the Configuration tab and selecting "Import" from the Console menu.

   The Import Database dialog box is displayed.

2. In the "LDIF file" field, enter the full path to the LDIF file you want to import, or click Browse to select the file you want to import.

   If you are running the Console on a machine remote to the directory, the field name appears as "LDIF file (on the machine running the Console)." This reminds you that when you do a browse, you are not browsing your current directory. Instead, you are browsing the filesystem of the machine running the Console.

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. If you want the server to ignore operations other than add, select the "Add only" checkbox.
   - **Continue on Error.** Select the "Continue on error" checkbox if you want the server to continue with the import even if errors occur. For example, you might use this option if you are importing 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 you want the server to record all entries it cannot import, or click Browse to select the file which will contain the rejects.

   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.

   If you leave this field blank, the server will not record rejected entries.

5. Click OK.

   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.

Initializing a Database from the Console

You can overwrite the existing data in a database. The following section describes using the Console to initialize databases.

You must be logged in as the Directory Manager in order to initialize a database. This is because you cannot import an LDIF file that contains a root entry unless you bind to the directory as the Directory Manager (Root DN). Only the Directory Manager has access to the root entry (for example, the root entry might be `dc=example,dc=com`).

CAUTION When initializing databases from an LDIF file, be careful not to overwrite the `o=NetscapeRoot` suffix unless you are restoring data. Otherwise, you will delete information that will require the reinstallation of the Directory Server.

To initialize a database using the Directory Server Console:

1. In the Directory Server Console, select the Configuration tab.
2. Expand the Data tree in the left navigation pane. Expand the suffix of the database you want to initialize, then click the database itself.
3. Right-click the database, and select Initialize Database.

   You can also select Initialize Database from the Object menu.
4. In the "LDIF file" field, enter the full path to the LDIF file you want to import, or click Browse to locate it on your machine.
5. If you are operating the Console from a machine local to the file being imported, skip to step 6. If you are operating the Console from a machine remote to the server containing the LDIF file, select one of the following options:
   - **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. By default, the Console looks for the file in the following directory:

     serverRoot`/slapd-`*serverID*`/ldif`
6. Click OK.

Importing from the Command-Line

You can use three methods for importing data through the command-line:

- Using `ldif2db` — This import method overwrites the contents of your database and requires the server to be stopped.
- Using `ldif2db.pl` — This import method overwrites the contents of your database while the server is still running.
- Using `ldif2ldap` — This method appends your LDIF file through LDAP. You can use this method to append data to all of your databases.

NOTE To import a database that has been encrypted, you must use the `-E` option with the script. See "Exporting and Importing an Encrypted Database," on page 102, for more information.

Importing Using the ldif2db Command-Line Script

The `ldif2db` script overwrites the data in a database you specify. The script requires you to shut down the server before proceeding with the import.

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.

CAUTION This script overwrites the data in your database.

To import LDIF with the server stopped:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Stop the server by typing the following:

   ```
   ./stop-slapd
   ```

3. Run the `ldif2db` command-line script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

CAUTION If you a specify a database in the `-n` option that 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 you do not misspell the database name.

An example of performing an import using the `ldif2db` UNIX shell script follows:

```
ldif2db -n Database1
 -i /opt/redhat-ds/servers/slapd-dirserver/ldif/demo.ldif
 -i /opt/redhat-ds/servers/slapd-dirserver/ldif/demo2.ldif
```

The following table describes the `ldif2db` options used in the examples:

Option	Description
`-i`	Specifies the full path name of the LDIF file(s) to be imported. This option is required. You can use multiple `-i` arguments to import more than one LDIF file at a time. When you import multiple files, the server imports the LDIF files in the order in which you specify them from the command-line.
`-n`	Specifies the name of the database into which you are importing the data.

For more information about using this script, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

Importing Using the ldif2db.pl Perl Script

As with the `ldif2db` script, the `ldif2db.pl` script overwrites the data in a database you specify. This script requires the server to be running in order to perform the import.

CAUTION This script overwrites the data in your database.

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Run the `ldif2db.pl` perl script.

   For more information about using this Perl script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

The following example imports an LDIF file using the `ldif2db.pl` script. You do not need `root` privileges to run the script, but you must authenticate as the Directory Manager.

UNIX shell script:

```
ldif2db.pl -D "cn=Directory Manager" -w secretpwd
 -i /opt/redhat-ds/servers/slapd-dirserver/ldif/demo.ldif
 -n Database1
```

The following table describes the `ldif2db.pl` options used in the examples:

Option	Description
`-D`	Specifies the DN of the administrative user.
`-w`	Specifies the password of the administrative user.
`-i`	Specifies the LDIF file(s) to be imported. This option is required. You can use multiple `-i` arguments to import more than one LDIF file at a time. When you import multiple files, the server imports the LDIF files in the order in which you specify them from the command-line.
`-n`	Specifies the name of the database into which you are importing the data.

Importing Using the ldif2ldap Command-Line Script

The `ldif2ldap` script appends the LDIF file through LDAP. Using this script, you import data to all directory databases at the same time. The server must be running in order to import using `ldif2ldap`.

To import LDIF using `ldif2ldap`:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Run the `ldif2ldap` command-line script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

An example of performing an import using the `ldif2ldap` UNIX shell script follows:

```
ldif2ldap "cn=Directory Manager" secret
/opt/redhat-ds/servers/slapd-dirserver/ldif/demo.ldif
```

The `ldif2ldap` script requires you to specify the DN of the administrative user, the password of the administrative user, and the absolute path and filename of the LDIF file(s) to be imported.

Exporting Data

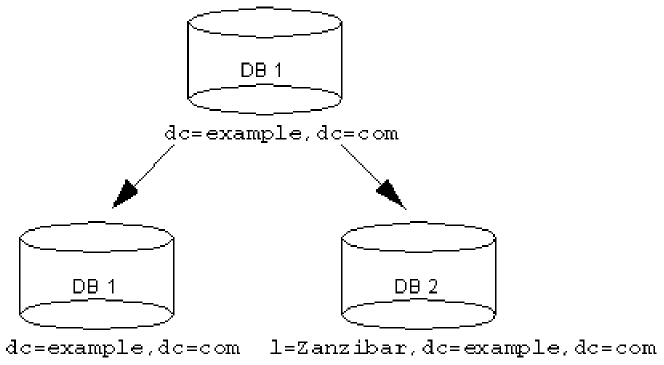
You can use the LDAP Data Interchange Format (LDIF) to export database entries from your databases. LDIF is a standard format described in RFC 2849, "The LDAP Data Interchange Format (LDIF) - Technical Specification."

Exporting data can be useful for the following:

- Backing up the data in your database.
- Copying your data to another Directory Server.
- Exporting your data to another application.
- Repopulating databases after a change to your directory topology.

Suppose your directory is contained by one database, and you decide to split its contents over two databases, as illustrated in Figure 4-1.

Figure 4-1 Splitting a Database Contents into Two Databases



To populate the new databases requires exporting the contents of database one and importing it into the new databases one and two.

You can use the Directory Server Console or command-line utilities to export data. The following sections describe these methods in detail:

- Exporting Directory Data to LDIF Using the Console
- Exporting a Single Database to LDIF Using the Console
- Exporting to LDIF from the Command-Line

The export operations do not export the configuration information (`cn=config`).

CAUTION Do not stop the server during an export operation.

Exporting Directory Data to LDIF Using the Console

You can export some or all of your directory data to LDIF, depending upon the location of the final exported file. When the LDIF file is on the server, you can export only the data contained by the databases local to the server. If the LDIF file is remote to the server, you can export all of the databases and database links.

To export directory data to LDIF from the Directory Server Console while the server is running:

1. In the Directory Server Console, select the Tasks tab. Scroll to the bottom of the screen, and click Export Database(s).

   To export all of your databases, you can also select the Configuration tab and select Export from the Console menu.

   The Export Database dialog box is displayed.

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 you are running the Console on a remote server. When the Browse button is not enabled, the file is stored by default in this directory:

   serverRoot`/slapd-`*serverID*`/ldif`

3. If you are running the Console on a machine remote to the server, two radio buttons are displayed beneath the LDIF file field.

- Select "To local machine" to indicate that you are exporting to an LDIF file in the machine from which you run the Console.
- Select "To server machine" to indicate that you are exporting to an LDIF file located on the server's machine.

4. If you want to export the whole directory, select the "Entire database" radio button.

   If you want 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 allows you to export a subtree that is contained by more than one database.

   You can also click Browse to select a suffix or subtree.

5. Click OK to export the file.

Exporting a Single Database to LDIF Using the Console

To export one database to LDIF from the Directory Server Console while the server is running:

1. In the Directory Server Console, select the Configuration tab.
2. Expand the Data tree in the left navigation pane. Expand the suffix maintained by the database you want to export. Select the database under the suffix that you want to export.
3. Right-click the database, and select Export Database.

   You can also select Export Database from the Object menu.

   The Export Partition dialog box is displayed.

4. In the "LDIF file" field, enter the full path to the LDIF file, or click Browse to locate it on your machine.

   When the Browse button is not enabled, by default the file is stored in this directory:

   serverRoot`/slapd-`*serverID*`/ldif`

5. Click OK to export the file.

Exporting to LDIF from the Command-Line

You can export your database to LDIF using the `db2ldif` command-line script. This script exports all of your database contents or a part of their contents to LDIF when the server is running or stopped.

NOTE To export a database that has been encrypted, you must use the `-E` option with the script. See "Exporting and Importing an Encrypted Database," on page 102, for more information.

To export to LDIF from the command-line:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Run the `db2ldif` command-line script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example performs an import using the `db2ldif` UNIX shell script:

```
db2ldif -n database1 -a output.ldif -s "dc=example,dc=com" -s
 "o=NetscapeRoot"
```

The following table describes the `db2ldif` options used in the examples:

Option	Description
`-n`	Specifies the name of the database from which the file is being exported.
`-a`	Defines the output file in which the server saves the exported LDIF. This file is stored by default in the directory where the command-line script resides.
`-s`	Specifies the suffix or suffixes to include in the export. You may use multiple `-s` arguments.

Backing Up and Restoring Data

You can back up and restore your databases using the Directory Server Console or a command-line script.

The following sections describe the procedures for backing up and restoring data:

- Backing Up All Databases
- Backing Up the dse.ldif Configuration File
- Restoring All Databases
- Restoring a Single Database
- Restoring Databases That Include Replicated Entries
- Restoring the dse.ldif Configuration File

CAUTION Do not stop the server during a backup or restore operation.

Backing Up All Databases

The following procedures describe backing up all of the databases in your directory using the Directory Server Console and from the command-line.

NOTE You cannot use these backup methods to back up the data contained by databases on a remote server that you chain to using database links.

Backing Up All Databases from the Server Console

When you back up your databases from the Directory Server Console, the server copies all of the database contents and associated index files to a backup location. You can perform a backup while the server is running.

To back up your databases from theDirectory Server Console:

1. In the Directory Server Console, select the Tasks tab.
2. Click Back Up Directory Server.

   The Backup Directory dialog box is displayed.

3. Enter the full path of the directory where you want 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 you are running the Console on the same machine as the directory, you can also click Browse to locate a local directory.

   If you choose to use the default, the backup files will be placed in the following location:

   serverRoot`/slapd-`*serverID*`/bak/`*backup_directory*

   The *backup_directory* variable names a directory using the name of the backup file. By default, the backup directory name contains the time and date the backup was created (`YYYY_MM_DD_hhmmss`).

4. Click OK to create the backup.

Backing Up All Databases from the Command-Line

You can back up your databases from the command-line using the `db2bak` command-line script. This script works when the server is running or when the server is stopped.

You cannot back up the configuration information using this backup method. For information on backing up the configuration information, refer to "Backing Up the dse.ldif Configuration File," on page 164.

To back up your directory from the command-line using the `db2bak` script:

1. At the command prompt, change to the following directory:

   serverRoot`/slapd-`*serverID*

2. Run the `db2bak` command-line script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example performs an import using the `db2bak` UNIX shell script:

```
db2bak /opt/redhat-ds/servers/slapd-dirserver/bak/bak_2001070110
3056
```

You can specify the backup directory and output file where the server saves the exported LDIF file. If you do not specify a directory and output file, the directory will store the file by default in the directory where the command-line script resides. By default, the backup file is named according to the year-month-day-hour format (`YYYY_MM_DD_hhmmss`).

Backing Up the dse.ldif Configuration File

Directory Server automatically backs up the `dse.ldif` configuration file. When you start your Directory Server, the directory creates a backup of the `dse.ldif` file automatically in a file named `dse.ldif.startOK` in this directory:

serverRoot`/slapd-`*serverID*`/config`

When you make modifications to the `dse.ldif` file, the file is first backed up to a file called `dse.ldif.bak` in the *serverRoot*`/slapd-`*serverID*`/config` directory before the directory writes the modifications to the `dse.ldif` file.

Restoring All Databases

The following procedures describe restoring all of the databases in your directory using the Directory Server Console and from the command-line.

NOTE While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Restoring All Databases from the Console

If your databases become corrupted, you can 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.

CAUTION Restoring your databases overwrites any existing database files.

To restore your databases from a previously created backup:

1. In the Directory Server Console, select the Tasks tab.
2. Click Restore Directory Server.

   The Restore Directory dialog box is displayed.

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:

   serverRoot/`slapd-`*serverID*/`bak`/*backup_name*

   where *backup_name* is the name of the backup file.

4. Click OK to restore your databases.

Restoring Your Database from the Command-Line

You can restore your databases from the command-line by using the following scripts:

- 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.

Using the bak2db Command-Line Script

To restore your directory from the command-line while the server is shut down:

1. At the command prompt, change to the following directory:

   serverRoot/`slapd-`*serverID*

2. If the server is running, type the following to stop it:

   ```
   ./stop-slapd
   ```

3. Run the `bak2db` command-line script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example performs an import using the `bak2db` UNIX shell script:

```
bak2db
/opt/redhat-ds/servers/slapd-dirserver/bak/bak_20010701103056
```

The `bak2db` script requires that you define the full path and name of the input file.

Using bak2db.pl Perl Script

To restore your directory from the command-line while the server is running:

1. At the command prompt, change to the following directory:

   serverRoot/`slapd-`*serverID*

2. Run the `bak2db.pl` Perl script.

   For more information on using this Perl script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example performs an import using the `bak2db.pl` UNIX shell script:

```
bak2db.pl -D "cn=Directory Manager" -w secret
 -a /opt/redhat-ds/servers/slapd-dirserver/bak/mybak_2001070110
3056
```

The following table describes the `bak2db.pl` options used in the examples:

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.

Restoring a Single Database

To restore a single database:

1. At the command prompt, change to the following directory:

   ```
   cd serverRoot/slapd-serverID
   ```

2. Stop the Directory Server if it is running by typing the following:

   ```
   ./stop-slapd
   ```

3. Restore the backend from your `/bak` archives with the `bak2db` script, using the `-n` parameter to specify the backend instance name. For example:

   ```
   ./bak2db /redhat/servers/slapd-serverID/bak/backup_file -n
   userRoot
   ```

4. Restart the Directory Server by typing the following:

   ```
   ./start-slapd
   ```

Restoring Databases That Include Replicated Entries

If you are restoring a database that is supplying entries to other servers, then you must reinitialize all of the servers that receive updates from the restored database (for example, consumer servers, hub servers, and, in multi-master replication environments, other supplier servers). 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.If you are restoring a database containing data received from a supplier server, then one of two situations can occur:

- Changelog entries have not yet expired on the supplier server.

  If the supplier server changelog has not expired since the database backup was taken, then you can 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 you have set for the maximum changelog age attribute. This attribute is called `nsslapd-changelogmaxage` and can be found in the `cn=changelog5,cn=config` entry. For more information about this option, refer to the *Red Hat Directory Server Configuration, Command, and File 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, you need to initiate consumer reinitialization. For more information on reinitializing consumers, refer to "Initializing Consumers," on page 350.

For information on managing replication, see "Managing Replication," on page 307.

Restoring the dse.ldif Configuration File

To restore the `dse.ldif` configuration file, stop the server, then use the procedure outlined in "Restoring a Single Database," on page 166, to copy the backup copy of the `dse.ldif` file into your directory. Once you have copied the data, restart your server.

The directory creates two backup copies of the `dse.ldif` file in this directory:

serverRoot`/slapd-`*serverID*`/config`

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. Copy the file with the most recent changes to your directory.

Enabling and Disabling Read-Only Mode

Before performing certain operations of export or backup on your Directory Server, you can enable read-only mode on any of the databases to ensure you have 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, if you have a multi-master configuration, this might not be a problem for you.

Enabling Read-Only Mode

1. In the Directory Server Console, select the Configuration tab, and expand the Data folder in the navigation tree.
2. Select the database that you want to place in read-only mode, and click the Database Settings tab in the right pane.
3. Select the “Database is Read-Only” checkbox.
4. Click Save.

   Your change takes effect immediately.

Before performing an import or restore operation, you should ensure that the databases affected by the operation are *not* in read-only mode. If they are, use the following procedure to make them available for updates.

Disabling Read-Only Mode

1. In the Directory Server Console, select the Configuration tab, and expand the Data tree.

2. Select the database that you want to make available for updates, and click the Database Settings tab in the right pane.
3. Clear the "Database is Read-only" checkbox.
4. Click Save.

   Your change takes effect immediately.

Chapter 5

Advanced Entry Management

You can group the entries contained within your directory to simplify the management of user accounts. Red Hat Directory Server (Directory Server) supports a variety of methods for grouping entries and sharing attributes between entries.

This chapter describes the following grouping mechanisms and their procedures:

- Using Groups (page 171)
- Using Roles (page 174)
- Assigning Class of Service (page 186)

To take full advantage of the features offered by roles and class of service, determine your directory topology in the planning phase of your directory deployment. Refer to the *Red Hat Directory Server Deployment Guide* for more information.

Using Groups

Groups are a mechanism for associating entries for ease of administration. This mechanism was provided with previous versions of Directory Server and should be used primarily for compatibility with older versions of the server.

The following sections describe managing static and dynamic groups. For a conceptual overview of groups, refer to the *Red Hat Directory Server Deployment Guide*. For more information about administering groups, refer to *Managing Servers with Red Hat Console*.

Managing Static Groups

Static groups allow you to group entries by specifying the same group value in the DN attribute of any number of users. This section includes the following procedures for creating and modifying static groups:

- Adding a New Static Group
- Modifying a Static Group

NOTE If you have a user that is remote from the definition of the static group, then you can use the Referential Integrity Plug-in to ensure that deleted user entries are automatically deleted from the static group.

For more information about using referential integrity with chaining, refer to "Configuring the Chaining Policy," on page 104.

Adding a New Static Group

1. In the Directory Server Console, select the Directory tab.
2. In the left pane, right-click the entry under which you want to add a new group, and select New > Group.

   You can also go to the Object menu and select New > Group.
3. Click General in the left pane. Type a name for your new group in the "Group Name" field.

   The group name is required.
4. Enter a description of the new group in the "Description" field.
5. Click Members in the left pane. In the right pane, select the Static Group tab. Click Add to add new members to the group.

   The standard "Search users and groups" dialog box appears.
6. In the Search drop-down list, select what sort of entries to search for (users, groups, or both) then click Search. Select one of the entries returned, and click OK.
7. Click Languages in the left pane to add language-specific information for your group.
8. Click OK to create your new group. It appears in the right pane.

Modifying a Static Group

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

   The directory contents appear in the left pane.

2. Double-click the entry you want to modify, or select Open from the Object menu.

   The Edit Group dialog box appears.

3. Make your changes to the group information. Click OK.

   To view your 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.

You may notice this problem 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 filter `(objectclass=person)` and scope `sub-tree`.

Managing Dynamic Groups

Dynamic groups filter users based on their DN and include them in a single group. This section contains the following procedures for creating and modifying dynamic groups:

- Adding a New Dynamic Group
- Modifying a Dynamic Group

Adding a New Dynamic Group

1. Follow steps 1-4 of “Adding a New Static Group,” on page 172.

2. 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.

   The standard “Construct and Test LDAP URL” dialog box displays.

3. Enter an LDAP URL in the text field or select Construct to be guided through the construction of an LDAP URL.

4. Click Languages in the left pane to add language-specific information for your group.
5. Click OK to create your new group.

   Your new group appears in the right pane.

Modifying a Dynamic Group

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

   The directory contents appear in the left pane.
2. Double-click the entry you want to modify, or select Properties from the Object menu.

   The Edit Group dialog box appears.
3. Make your changes to the group information. Click OK.

   To view your changes, 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.

You may notice this problem 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 filter `(objectclass=person)` and scope `sub-tree`.

Using Roles

Roles are a new 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 locate the role of an entry, rather than select a group and browse the members list.

This section contains the following topics:

- About Roles
- Managing Roles Using the Console
- Managing Roles Using the Command-Line

- Using Roles Securely

About Roles

Roles unify the static and dynamic group concept supported by previous versions of Directory Server.

You can use roles to:

- Enumerate the members of a role.

  Having an enumerated list of role members can be useful for resolving queries for role members quickly.

- Determine whether a given entry possesses a particular role.

  Knowing the roles possessed by an entry can help you determine whether the entry possesses the target role.

- Enumerate all the roles possessed by a given entry.
- Assign a particular role to a given entry.
- Remove a particular role from a given entry.

With managed roles, you can do everything you would normally do with static groups, and you can filter members using filtered roles as you used to do with dynamic groups. Roles are easier to use than groups, more flexible in their implementation, and reduce client complexity.

However, evaluating roles is more resource-intensive because the server does the work for the client application. With roles, the client application can check role membership by searching 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.

NOTE Use the `nsRole` attribute, not the `nsRoleDN` attribute, to evaluate role membership.

Each role has *members*, or entries that possess the role. You can specify members either explicitly or dynamically. How you specify role membership depends upon the type of role you are using. Directory Server supports three types of roles:

- Managed roles — A managed role allows you to create an explicit enumerated list of members.
- Filtered roles — A filtered role allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.
- Nested roles — A nested role allows you to create roles that contain other roles.

For more information about how roles work, refer to *Red Hat Directory Server Deployment Guide*.

The concept of activating/inactivating roles is introduced to enable you to activate/inactivate groups of entries in just one operation. That is, you can temporarily disable the members of a role by inactivating the role to which they belong.

When a role is said to be inactivated, it does not mean that you cannot bind to the server using that role entry. The meaning of an inactivated role is that you 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`.

In the case of the nested role, an inactivated nested role means that you cannot bind to the server using an entry that belongs to a role that is a member of the nested role. All the entries that belong to a role that directly or indirectly are members of the nested role (one may have several levels of nested roles) will have `nsaccountlock` set to `true`.

Managing Roles Using the Console

This section contains the following procedures for creating and modifying roles:

- Creating a Managed Role
- Creating a Filtered Role
- Creating a Nested Role
- Viewing and Editing an Entry's Roles
- Modifying a Role Entry
- Making a Role Inactive
- Reactivating a Role

- Deleting a Role

When you create a role, you need to decide whether a user can add themselves or remove themselves from the role. Refer to "Using Roles Securely," on page 184, for more information about roles and access control.

Creating a Managed Role

Managed roles allow you to create an explicit enumerated list of members. Managed roles are added to entries by adding the `nsRoleDN` attribute to the entry.

To create and add members to a managed role:

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 your new role.
3. Go to the Object menu, and select New > Role.

   You can also right click the entry and select New > Role.

   The Create New Role dialog box is displayed.
4. Click General in the left pane. Type a name for your 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 Managed Role. Click Add to add new entries to the list of members.

   The standard "Search users and groups" dialog box appears.
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. When you have finished adding entries to the role, click OK.

   The new role appears in the right pane.

Creating a Filtered Role

You assign entries to a filtered role depending upon a particular attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

To create and add members to a filtered role:

1. Follow steps 1-5 of "Creating a Managed Role," on page 177.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Filtered Role.
4. Enter an LDAP filter in the text field, or click Construct to be guided through the construction of an LDAP filter.
5. If you click Construct, the standard LDAP URL construction dialog appears. Disregard the LDAP Server Host, Port, Base DN, and Search (as you cannot specify a search scope for filtered role definitions) fields.
   a. Select the types of entries you want to filter from the "For" drop-down list.

      You can choose between users, groups, or both.
   b. Select an attribute from the "Where" drop-down list. The two fields following it allow you to refine your search by selecting one of the qualifiers from the drop-down list (such as contains, does not contain, is, is not) and enter an attribute value in the text box. To add additional filters, click More. To remove unnecessary filters, click Fewer.
   c. Click OK to save your filter.
6. Click Test to try your filter.

   A Filter Test Result dialog box displays the entries matching your filter.
7. Click OK.

   The new role appears in the right pane.

Creating a Nested Role

Nested roles allow you to create roles that contain other roles. Before you create a nested role, another role must exist. When you create a nested role, the Console displays a list of the roles available for nesting. The roles nested within the nested role are specified using the `nsRoleDN` attribute.

To create and add members to a nested role:

1. Follow steps 1-5 of “Creating a Managed Role,” on page 177.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Nested Role.
4. Click Add to add roles to the list.The members of the nested role are members of other existing roles.

   The Role Selector dialog box appears.
5. Select a role from the “Available roles” list, and click OK.
6. Click OK.

   The new role appears in the right pane.

Viewing and Editing an Entry’s Roles

To view or edit a role associated with an entry from the Console:

1. In the Directory Server Console, select the Directory tab.
2. In the left navigation pane, browse the tree, and select the entry for which you want to view or edit a role.
3. Select Set Roles from the Object menu.

   The Roles dialog box displays.
4. Select the Managed Roles tab to display the managed roles to which this entry belongs.

   To add a new managed role, click Add, and select an available role from the Role Selector window. Click OK.

   To remove a managed role, select it, and click Remove.

   To edit a managed role associated with an entry, click Edit. The Edit Entry dialog box displays. Make any changes to the general information or members and click OK.
5. 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. Click OK to save your changes.

6. Click OK to save your changes once you have finished modifying the roles.

Modifying a Role Entry

To edit an existing role:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Double-click the role.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the role name and description.
5. Click Members in the left pane to change the members of managed and nested roles or to change the filter of a filtered role.
6. Click OK to save your changes.

Making a Role Inactive

You can temporarily disable the members of a role by inactivating the role to which they belong. Inactivating a role inactivates the entries possessed by the role, not the role itself.

To temporarily disable the members of a role:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Select the role. Select Inactivate from the Object menu.

   You can also right-click the role and select Inactivate from the menu.

   The role is inactivated.

   To see the inactivated entries, select Inactivation State from the View menu. A red slash through the role icon indicates that the role has been inactivated.

Reactivating a Role

To reactivate a disabled role:

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

2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.

3. Select the role. Select Activate from the Object menu.

   You can also right-click the role and select Activate from the menu.

   The role is reactivated.

   To see inactivated entries, select Inactivation State from the View menu.The role icon appears as normal, indicating that the role is active.

Deleting a Role

Deleting a role deletes the role only, not its members.

To delete a role:

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

2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.

3. Right-click the role, and select Delete.

   A dialog box appears asking you to confirm the deletion. Click Yes.

4. The Deleted Entries dialog box appears to inform you that the role was successfully deleted. Click OK.

NOTE Deleting a role deletes the role entry but does not delete the `nsRoleDN` attribute for each role member. If you want 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 “Maintaining Referential Integrity,” on page 75.

Managing Roles Using the Command-Line

Roles inherit from the `ldapsubentry` object class, which is defined in the ISO/IEC X.509 standard. In addition, each type of role has two specific object classes that inherit from the `nsRoleDefinition` object class. Once you create a role, you assign members to it as follows:

- Members of a managed role have the `nsRoleDN` attribute in their entry.

- Members of a filtered role are entries that match the filter specified in the `nsRoleFilter` attribute.
- Members of a nested role are members of the roles specified in the `nsRoleDN` attributes of the nested role definition entry.

Table 5-1 lists the new object classes and attributes associated with each type of role.

Table 5-1 Object Classes and Attributes for Roles

Role Type	**Object Classes**	**Attributes**
Managed Role	`nsSimpleRoleDefinition` `nsManagedRoleDefinition`	Description (optional)
Filtered Role	`nsComplexRoleDefinition` `nsFilteredRoleDefinition`	`nsRoleFilter` Description (optional)
Nested Role	`nsComplexRoleDefinition` `nsNestedRoleDefinition`	`nsRoleDN` Description (optional)

NOTE In some cases, you need to protect the value of the `nsRoleDN` attribute with an ACI, as the attribute is writable. For more information about security and roles, refer to "Using Roles Securely," on page 184.

Examples: Managed Role Definition

You want to create a role to be assigned to all marketing staff. Run the `ldapmodify` script as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

Specify the managed role as follows:

```
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
```

Notice that the nsManagedRoleDefinition object class inherits from the LDAPsubentry, nsRoleDefinition, and nsSimpleRoleDefinition object classes.

Assign the role to a marketing staff member named Bob by doing an ldapmodify as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
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 present in the entry indicates that the entry is a member of a managed role, the marketing managed role cn=Marketing,ou=people,dc=example,dc=com.

Example: Filtered Role Definition

You want to set up a filtered role for sales managers. Run the ldapmodify script as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

Specify the filtered role as follows:

```
dn: cn=SalesManagerFilter,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: SalesManagerFilter
nsRoleFilter: o=sales managers
Description: filtered role for sales managers
```

Notice that the nsFilteredRoleDefinition object class inherits from the LDAPsubentry, nsRoleDefinition, and nsComplexRoleDefinition object classes. The nsRoleFilter attribute specifies the o (organization) attributes that contain the value of 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:

```
dn: cn=Pat,ou=people,dc=example,dc=com
objectclass: person
cn: Pat
sn: Pat
userPassword: bigsecret
o: sales managers
```

Example: Nested Role Definition

You want to create a role that contains both the marketing staff and sales managers contained by the roles you created in the previous examples. The nested role you created using `ldapmodify` appears as follows:

```
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
```

Notice the `nsNestedRoleDefinition` object class inherits from the `LDAPsubentry`, `nsRoleDefinition`, and `nsComplexRoleDefinition` object classes. The `nsRoleDN` attributes contain the DN of the marketing managed role and the sales managers filtered role.

Both of the users in the previous examples, Bob and Pat, would be members of this new nested role.

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 you had an interest group role called `Mountain Biking`, you would want interested users to add themselves or remove themselves easily.

However, in some security contexts, it is inappropriate to have such open roles. Consider account inactivation roles. By default, account inactivation roles contain ACIs defined for their suffix. When creating a role, the server administrator decides whether a user can assign themselves to or remove themselves from the role.

For example, user A possesses the managed role, MR. The MR role has been locked using account inactivation through the command-line. This means that user A cannot bind to the server because the `nsAccountLock` attribute is computed as `true` for that user. However, suppose the user was already bound and noticed that he is now locked through the MR role. If there are no ACIs preventing him, the user can remove the `nsRoleDN` attribute from his entry and unlock himself.

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")
  (targattrfilters="

  add=nsRoleDN:(!(nsRoleDN=cn=AdministratorRole,dc=example,dc
=com)),

  del=nsRoleDN:(!(nsRoleDN=cn=nsManagedDisabledRole,dc=exampl
e,dc=com))")
(version3.0;aci "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 the above points should be considered for each of the roles that comprise the nested role.

For more information about account inactivation, see "Inactivating Users and Roles," on page 300.

Assigning Class of Service

A class of service (CoS) allows you to share attributes between entries in a way that is transparent to applications. CoS simplifies entry management and reduces storage requirements.

There are two methods for creating and managing CoS: with Directory Server Console or through the command-line. The following sections describe CoS in more detail and provide the procedures for managing CoS through both the Console and the command-line:

- About CoS
- Managing CoS Using the Console
- Managing CoS from the Command-Line
- Creating Role-Based Attributes
- Access Control and CoS

About CoS

Clients of the Directory Server read the attributes on a user's entry. With CoS, some attribute values may not be stored with the entry itself. Instead, they are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of the following two types of entry in your directory:

- CoS Definition Entry — The CoS definition entry identifies the type of CoS you are using. 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.

NOTE LDAP search requests containing a filter that references an attribute defined by a CoS may return unexpected results. Take care when deciding which attributes to generate using a CoS.

The following sections describe the entries that make up a CoS in more detail and provide examples of each type of CoS.

About the CoS Definition Entry

The CoS definition entry is an instance of the `cosSuperDefinition` object class. The CoS definition entry also contains an object class that specifies the type of template entry it uses to generate the entry. You can specify three different object classes depending upon the type of CoS you want to use. The target entries 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, refer to "Managing CoS from the Command-Line," on page 194.

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, you can use the CoS definition entry to control this behavior.

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.

- 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.

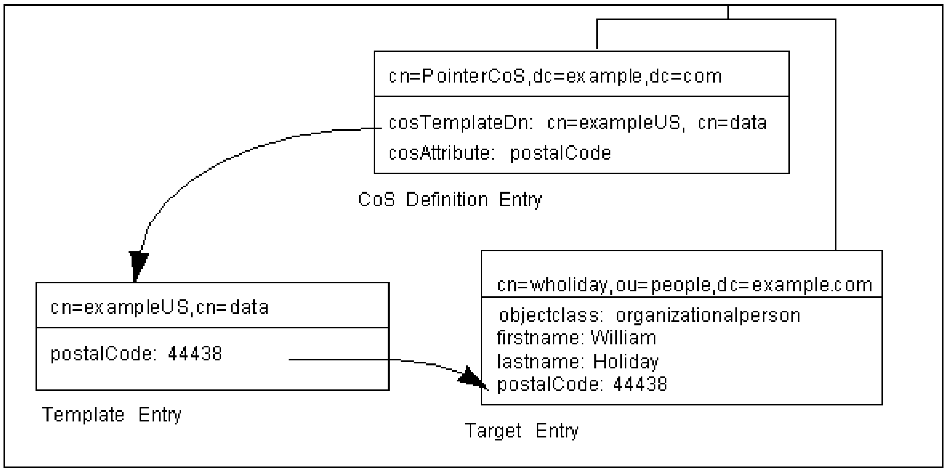
- By a combination of the DN of the subtree where the CoS performs a one level search for templates and the value of one of the target entry's attributes.

  This type of template is associated with a classic CoS.

How a Pointer CoS Works

You create 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.

Figure 5-1 Sample Pointer CoS



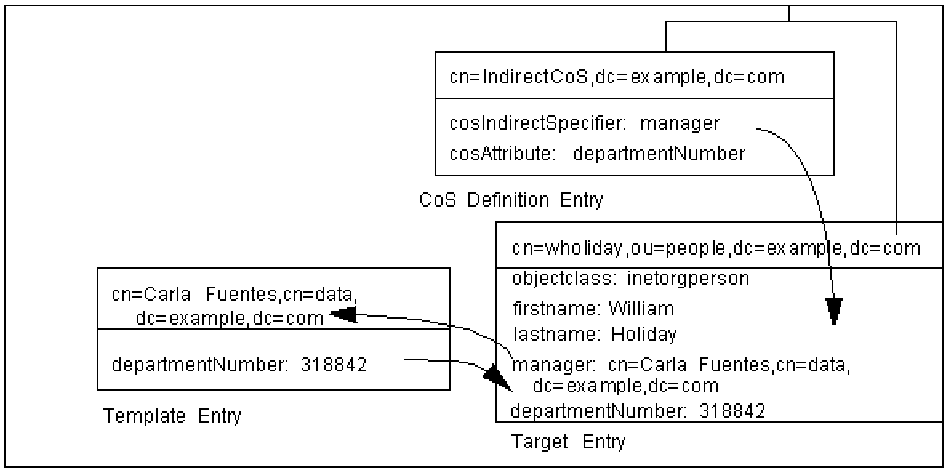
In this example, the template entry is identified by its DN, `cn=exampleUS,cn=data`, in the CoS definition entry. Each time the `postalCode` attribute is queried on the entry `cn=wholiday,ou=people,dc=example,dc=com`, the Directory Server returns the value available in the template entry `cn=exampleUS,cn=data`.

How an Indirect CoS Works

You can create 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.

Figure 5-2 Sample Indirect CoS


cn=IndirectCoS,dc=example,dc=com
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
CoS Definition Entry
cn=wholiday,ou=people,dc=example,dc=com
objectclass: inetorgperson
firstname: William
lastname: Holiday
manager: cn=Carla Fuentes,cn=data, dc=example,dc=com
departmentNumber: 318842
Target Entry
cn=Carla Fuentes,cn=data, dc=example,dc=com
departmentNumber: 318842
Template Entry

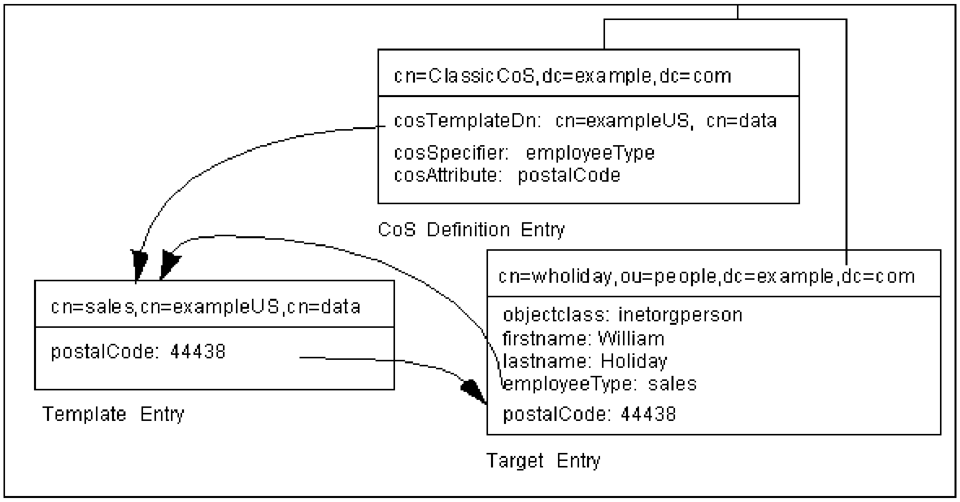
In this example, the target entry for William Holiday contains the indirect specifier, the `manager` attribute. William's manager is Carla Fuentes, so the `manager` attribute contains a pointer to the DN of the template entry, `cn=Carla Fuentes,ou=people,dc=example,dc=com`. The template entry in turn provides the `departmentNumber` attribute value of `318842`.

How a Classic CoS Works

You can create 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:

Figure 5-3 Sample Classic CoS


cn=ClassicCoS,dc=example,dc=com
cosTemplateDn: cn=exampleUS, cn=data
cosSpecifier: employeeType
cosAttribute: postalCode
CoS Definition Entry
cn=sales,cn=exampleUS,cn=data
postalCode: 44438
Template Entry
cn=wholiday,ou=people,dc=example,dc=com
objectclass: inetorgperson
firstname: William
lastname: Holiday
employeeType: sales
postalCode: 44438
Target Entry

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.

Managing CoS Using the Console

This section describes creating and editing CoS through the Directory Server Console. It includes the following sections:

- Creating a New CoS
- Creating the CoS Template Entry
- Editing an Existing CoS
- Deleting a CoS

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 your new class of service.

3. Go to the Object menu, and select New > Class of Service.

   You can also right click the entry and select New > Class of Service.

   The Create New Class of Service dialog displays.

4. Select General in the left pane. In the right pane, enter the name of your 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. Once you have added an attribute 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 You can only make an attribute operational if it is also defined as operational in the schema.

For example, if your 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.** If you choose 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 your local server. This will be an exact DN; e.g., `cn=CoS template,ou=People,dc=example,dc=com`.
- **Using the value of one of the target entry’s attribute.** If you choose 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.** If you choose 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; set the suffix or subsuffix where you will place the template entries (there can be more than one template).

8. Click OK.

Creating the CoS Template Entry

If you created a pointer CoS or a classic CoS, you need to create a template entry, according to the template DN you set when you created the class of service. Although you can place the template entries 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 you gave when you created the CoS. 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 your class of service.

   The CoS appears in the right pane with other entries.
3. Right-click on the CoS and select "New>Other."
4. Select `costemplate` from the list of object classes.

NOTE You can also add 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 (for example, if it is a user entry), you do not need to make it an instance of the `LDAPsubentry` object class.

5. The Properties Editor will appear.
6. Select the object classes attribute, and hit "Add Value." Add the extensibleobject object class. This will allow you to add any attribute available in the directory.
7. Click on the "Add Attribute" button. Add the `cn` attribute, and give it a value that will correspond to the attribute value in your target entry. For example, if you select the `manager` attribute to set the value for a classic CoS, give the `cn` a value of `uid=bparker,ou=people,dc=example,dc=com`. You can also set it to a role, such as `cn=QA Role,dc=example,dc=com` or a regular attribute value (if you selected the `employeetype` attribute, for example, it can be `full time` or `temporary`).
8. Click the "Add Attribute" button, and add the attributes that you listed in the CoS. The values you use here will be used throughout the directory in the targeted entries.
9. 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 are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily. The behavior for negative `cosPriority` values is not defined in Directory Server; do not enter negative values. Also, the `cosPriority` attribute is not supported by indirect CoS.
10. Hit save.

The CoS will be visible in the left navigation pane once there are entries beneath it. For classic CoS, you can add multiple entries, according to the different potential values of the attribute specifier you set.

Editing an Existing CoS

The following procedure describes changing the description and attributes generated on the target entry of an existing class of service.

To edit an existing 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 that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Double-click the CoS.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the CoS name and description.
5. Click Attributes in the left pane to add or remove attributes generated by the CoS.
6. Click OK to save your changes.

   The target entries of the CoS are automatically updated.

Deleting a CoS

The following procedure describes deleting a 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 that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Right-click the CoS, and select Delete. A dialog box appears asking you to confirm the deletion. Click Yes.
4. The Deleted Entries dialog box appears to inform you that the CoS was successfully deleted. Click OK.

Managing CoS from the Command-Line

Because all configuration information and template data is stored as entries in the directory, you can use standard LDAP tools for CoS configuration and management. This section contains the following topics:

- Creating the CoS Definition Entry from the Command-Line
- Creating the CoS Template Entry from the Command-Line
- Example of a Pointer CoS
- Example of an Indirect CoS
- Example of a Classic CoS

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. Table 5-2 lists the object classes associated with each type of CoS definition entry.

Table 5-2 CoS Definition Entry Object Classes

CoS Type	Object Classes	Description
Pointer CoS	`cosPointerDefinition`	Identifies the template entry associated with the CoS definition using the template entry's DN value. The DN of the template entry is specified in the `cosTemplateDn` attribute.
Indirect CoS	`cosIndirectDefinition`	Identifies the template entry using the value of one of the target entry's attributes. The attribute of the target entry is specified in the `cosIndirectSpecifier` attribute.
Classic CoS	`cosClassicDefinition`	Identifies the template entry using both the template entry's DN (as specified in the `cosTemplateDn` attribute) and the value of one of the target entry's attributes (as specified in the `cosSpecifier` attribute).

Table 5-3 lists attributes that you can use in your CoS definition entries.

Table 5-3 CoS Definition Entry Attributes

Attribute	Definition
`cosAttribute`	Provides the name of the attribute for which you want to generate a value. You can specify more than one `cosAttribute` value. This attribute is used by all types of CoS definition entries.

Table 5-3 CoS Definition Entry Attributes *(Continued)*

Attribute	Definition
`cosIndirectSpecifier`	Specifies the attribute value used by an indirect CoS to identify the template entry.
`cosSpecifier`	Specifies the attribute value used by a classic CoS, which, along with the template entry's DN, identifies the template entry.
`cosTemplateDn`	Provides the DN of the template entry associated with the CoS definition. Used for pointer CoS and classic CoS only.

The `cosAttribute` attribute allows an additional qualifier after the attribute value. You can use the following qualifiers:

- `Default`

  This qualifier indicates that the server only returns a generated value if there is no corresponding attribute value stored with the entry.

- `Override`

  This qualifier indicates that the server always returns the value generated by the CoS, even when there is a value stored with the entry.

- `Operational`

  This qualifier indicates that the attribute will only be returned if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order to be returned. When you use `operational` as a qualifier, it works as if `override` and `operational` were specified.

- `Operational-default`

  This qualifier indicates that the server 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.

NOTE You can only make an attribute operational if it is also defined as operational in the schema. For example, if your CoS generates a value for the `description` attribute, you use the `operational` qualifier because this attribute is not marked `operational` in the schema.

If you do not indicate a qualifier, `default` is assumed.

For example, you might create a pointer CoS definition entry that contains an `override` qualifier as follows:

```
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
```

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.

NOTE If an entry contains an attribute value generated by a CoS, you cannot manually update the value of the attribute if it is defined with the `operational` or `override` qualifiers.

For more information about the attributes, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

Now that you have been introduced to the object classes and attributes used by a CoS definition, it is time to put them together to create the definition entry itself. Table 5-4 describes the CoS definition for each type of CoS.

Table 5-4 CoS Definitions

CoS Type	CoS definition

Pointer CoS	`objectclass: top`
	`objectclass: cosSuperDefinition`
	`objectclass: cosPointerDefinition`
	`cosTemplateDn:` *DN_string*
	`cosAttribute:` *list_of_attributes qualifier*

Indirect CoS	`objectclass: top`
	`objectclass: cosSuperDefinition`
	`objectclass: cosIndirectDefinition`
	`cosIndirectSpecifier:` *attribute_name*
	`cosAttribute:` *list_of_attributes qualifier*

Table 5-4 CoS Definitions *(Continued)*

CoS Type	**CoS definition**

Classic CoS	`objectclass: top`
	`objectclass: cosSuperDefinition`
	`objectclass: cosClassicDefinition`
	`cosTemplateDn:` *DN_string*
	`cosSpecifier:` *attribute_name*
	`cosAttribute:` *list_of_attributes qualifier*

Creating the CoS Template Entry from the Command-Line

Each template entry is an instance of the `cosTemplate` object class.

NOTE You can also add 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 (for example, if it is a user entry), you do not need to make it an instance of the `LDAPsubentry` object class.

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
```

It is possible to create CoS templates that compete with each other to provide an attribute value. For example, you might have a multi-valued `cosSpecifier` in your CoS definition entry. In such a case, you can specify a template priority on each template entry to determine which template provides the attribute value. Set the template priority using the `cosPriority` attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

Templates that contain no cosPriority attribute are considered the lowest priority. In the case 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. The behavior for negative cosPriority values is not defined in Directory Server; do not enter negative values. Also, the cosPriority attribute is not supported by indirect CoS.

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.

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

Example of a Pointer CoS

You want to create a pointer CoS that shares a common postal code with all entries in the dc=example,dc=com tree.

To add a new pointer CoS definition entry to the dc=example,dc=com suffix, you do an ldapmodify as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The ldapmodify utility binds to the server and prepares it to add information to the configuration file.

Next, you add the pointer CoS definition to the dc=example,dc=com root suffix as follows:

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode
```

Next, you create the template entry as follows:

```
dn: cn=exampleUS,ou=data,dc=example,dc=com
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.

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.

First, you add a new indirect CoS definition entry to the `dc=example,dc=com` suffix, using `ldapmodify` as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add the indirect CoS definition to the `dc=example,dc=com` root suffix as follows:

```
dn: cn=indirectCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

You do not need to add any additional entries to the directory or modify the manager entries as long as they already contain the `departmentNumber` attribute. 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.

Example of a Classic CoS

You want to create a classic CoS that automatically generates postal codes using a combination of the template DN and the attribute specified in the `cosSpecifier` attribute.

First, you add a new classic CoS definition entry to the `dc=example,dc=com` suffix, using `ldapmodify` as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add the indirect CoS definition to the `dc=example,dc=com` root suffix, as follows:

```
dn: cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=classicCoS,dc=example,dc=com
cosSpecifier: businessCategory
cosAttribute: postalCode override
```

Next, you 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
objectclass: top
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438

dn: cn=marketing,cn=classicCoS,dc=example,dc=com
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 cosTemplate DN, 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.

Creating Role-Based Attributes

You can create classic CoS schemes that generate attribute values for an entry based on the role possessed by the entry. For example, you could use role-based attributes 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, you can define CoS schemes that have more than one possible template entry. To resolve the ambiguity of which template entry to use, you can include the `cosPriority` attribute in your CoS template entry.

For example, you can create a CoS that allows members of the manager role to exceed the standard mailbox quota. The manager role exists as follows:

```
dn: cn=ManagerRole,ou=people,dc=example,dc=com
objectclass: top
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: ManagerRole
nsRoleFilter: o=managers
Description: filtered role for managers
```

The classic CoS definition entry would look as follows:

```
dn: cn=managerCOS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectlass: 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
objectlass: 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.

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.

Chapter 6

Managing Access Control

Red Hat Directory Server (Directory Server) provides you with the ability to control access to your directory. This chapter describes the access control mechanism.

This section includes the following topics:

To take full advantage of the power and flexiblity of the access control mechanism, while you are in the planning phase for your directory deployment, you should define an access control strategy as an integral part of your overall security policy. Refer to *Red Hat Directory Server Deployment Guide* for tips on planning your access control strategy.

Access Control Principles

The mechanism by which you define access is called *access control*. When the server receives a request, it uses the authentication information provided by the user in the bind operation and the access control instructions (ACIs) defined in the server to allow or deny access to directory information. The server can allow or deny permissions such as read, write, search, and compare. The permission level granted to a user may be dependent on the authentication information provided.

Using access control, you can control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can set permissions for a specific user, all users belonging to a specific group or role, or all users of the directory. Finally, you can define access for a specific location such as an IP address or a DNS name.

ACI Structure

Access control instructions are stored in the directory as attributes of entries. The `aci` attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by the Directory Server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The `aci` attribute is returned in an `ldapsearch` operation if specifically requested.

The three main parts of an ACI statement are:

- Target
- Permission
- Bind Rule

The permission and bind rule portions of the ACI are set as a pair, also called an Access Control Rule (ACR). The specified permission is granted or denied depending on whether the accompanying rule is evaluated to be true.

ACI Placement

If an entry containing an ACI does not have any child entries, the ACI applies to that entry only. If the entry has child entries, the ACI applies to the entry itself and all entries below it. As a direct consequence, when the server evaluates access permissions to any given entry, it verifies the ACIs for every entry between the one requested and the directory suffix, as well as the ACIs on the entry itself.

The `aci` attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

You can create an ACI on an entry that does not apply directly to that entry but to some or all of the entries in the subtree below it. The advantage of this is that you can place at a high level in the directory tree a general ACI that effectively applies to entries more likely to be located lower in the tree. For example, at the level of an `organizationalUnit` entry or a `locality` entry, you could create an ACI that targets entries that include the `inetorgperson` object class.

You can use this feature to minimize the number of ACIs in the directory tree by placing general rules at high level branch points. To limit the scope of more specific rules, you should place them as close as possible to leaf entries.

NOTE ACIs placed in the root DSE entry apply only to that entry.

ACI Evaluation

To evaluate the access rights to a particular entry, the server compiles a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored on the Directory Server. ACIs are evaluated across all of the databases for a particular Directory Server but not across Directory Servers.

The evaluation of this list of ACIs is done based on the semantics of the ACIs, not on their placement in the directory tree. This means that ACIs that are close to the root of the directory tree do not take precedence over ACIs that are closer to the leaves of the directory tree.

The precedence rule that applies is that ACIs that deny access take precedence over ACIs that allow access. Between ACIs that allow access, union semantics apply, so there is no precedence.

For example, if you deny write permission at the directory's root level, then none of the users can write to the directory, regardless of the specific permissions you grant them. To grant a specific user write permissions to the directory, you have to restrict the scope of the original denial for write permission so that it does not include the user.

ACI Limitations

When creating an access control policy for your directory service, you need to be aware of the following restrictions:

- If your directory tree is distributed over several servers using the chaining feature, some restrictions apply to the keywords you can use in access control statements:
  - ACIs that depend on group entries (`groupdn` keyword) must be located on the same server as the group entry. If the group is dynamic, then all members of the group must have an entry on the server, too. If the group is static, the members' entries can be located on remote servers.
  - ACIs that depend on role definitions (`roledn` keyword) must be located on the same server as the role definition entry. Every entry that is intended to have the role must also be located on the same server.

  However, you can do value matching of values stored in the target entry with values stored in the entry of the bind user (for example, using the userattr keyword). Access will be evaluated normally even if the bind user does not have an entry on server that holds the ACI.

  For more information on how to chain access control evaluation, see "Database Links and Access Control Evaluation," on page 123.

- Attributes generated by a CoS cannot be used in all ACI keywords. Specifically, you should not use attributes generated by CoS with the following keywords:
  - `targetfilter` (see "Targeting Entries or Attributes Using LDAP Filters," on page 216)
  - `targattrfilters` (see "Targeting Attribute Values Using LDAP Filters," on page 217)
  - `userattr` (see "Using the userattr Keyword," on page 231)

If you create target filters or bind rules that depend on the value of attributes generated by CoS, the access control rule will not work. For more information on CoS, see chapter 5, "Advanced Entry Management."

- Access control rules are always evaluated on the local server. Therefore, it is not necessary to specify the hostname or port number of the server in LDAP URLs used in ACI keywords. If you do, the LDAP URL will not be taken into account at all. For more information on LDAP URLs, see Appendix C, "LDAP URLs."

Default ACIs

When you install the Directory Server, the following default ACIs apply to your directory information stored in the `userRoot` database:

- Users can modify a list of common attributes in their own entries. Those attributes include `mail`, `telephoneNumer`, `userPassword`, `seeAlso`, and so on. Operational and most of the security attributes, such as `aci`, `nsroledn`, and `passwordExpirationTime`, can't be modified by the users.
- Users have anonymous access to the directory for search, compare, and read operations.
- The administrator (by default `uid=admin,ou=Administrators, ou=TopologyManagement,o=NetscapeRoot`) has all rights except proxy rights.
- All members of the Configuration Administrators group have all rights except proxy rights.
- All members of the Directory Administrators group have all rights except proxy rights.
- SIE (Server Instance Entry) group.

Whenever you create a new database in the directory, the top entry has the default ACIs listed above.

The `NetscapeRoot` subtree has its own set of default ACIs:

- All members of the Configuration Administrators group have all rights on the `NetscapeRoot` subtree except proxy rights.
- Users have anonymous access to the `NetscapeRoot` subtree for search and read operations.
- Group expansion.

- All authenticated users have search, compare, and read rights to configuration attributes that identify the Administration Server.

The following sections explain how to modify these default settings to suit the needs of your organization.

Creating ACIs Manually

You can create access control instructions manually using LDIF statements and add them to your directory tree using the `ldapmodify` utility. The following sections explain in detail how to create the LDIF statements.

TIP LDIF ACI statements can be very complex. However, if you are setting access control for a large number of directory entries, using LDIF is the preferred method over using the Console because of the time it can save.

To familiarize yourself with LDIF ACI statements, however, you may want to use the Directory Server Console to set the ACI and then click the Edit Manually button on the Access Control Editor. This shows you the correct LDIF syntax. If your operating system allows it, you can even copy the LDIF from the Access Control Editor and paste it into your LDIF file.

The ACI Syntax

The `aci` attribute uses the following syntax:

```
aci: (target)(version 3.0;acl "name";permission bind_rules;)
```

where

- *target* specifies the entry, attributes, or set of entries and attributes for which you want to control access. The target can be a distinguished name, one or more attributes, or a single LDAP filter. The target is an optional part of the ACI.
- `version 3.0` is a required string that identifies the ACI version.
- `"`*name*`"` is a name for the ACI. The name can be any string that identifies the ACI. The ACI name is required.

- *permission* specifically outlines what rights you are either allowing or denying (for example, read or search rights).
- *bind_rules* specify the credentials and bind parameters that a user has to provide to be granted access. Bind rules can also specifically deny access to certain users or groups of users.

You can have multiple permission-bind rule pairs for each target. This allows you to set multiple access controls for a given target efficiently. For example:

target(*permission bind_rule*)(*permission bind_rule*)...

If you have several ACRs in one ACI statement, the syntax is of the form:

`aci: (`*target*`)(version 3.0;acl "`*name*`";`*permission bind_rule*`;` *permission bind_rule*`; ...` *permission bind_rule*`;)`

Example ACI

The following is an example of a complete LDIF ACI:

```
aci:
(target="ldap:///uid=bjensen,dc=example,dc=com")(targetattr=*)
(version 3.0;acl "aci1";allow (write) userdn="ldap:///self";)
```

In this example, the ACI states that the user `bjensen` has rights to modify all attributes in her own directory entry.

The following sections describe the syntax of each portion of the ACI in more detail.

Defining Targets

The target identifies to what the ACI applies. If the target is not specified, the ACI applies to the entry containing the `aci` attribute and to the entries below it.

A target can be:

- A directory entry or all of the entries in a subtree, as described in "Targeting a Directory Entry," on page 213.
- Attributes of an entry, as described in "Targeting Attributes," on page 215.
- A set of entries or attributes that match a specified LDAP filter, as described in "Targeting Entries or Attributes Using LDAP Filters," on page 216.

- An attribute value, or a combination of values, that match a specified LDAP filter, as described in "Targeting Attribute Values Using LDAP Filters," on page 217.

The general syntax for a target is:

(*keyword* = "*expression*")

(*keyword* != "*expression*")

where:

- *keyword* indicates the type of target.
- equal (=) indicates that the target is the object specified in the *expression*, and not equal (!=) indicates the target is not the object specified in the *expression*.
- *expression* identifies the target.

The quotation marks ("") around *expression* are required. What you use for *expression* is dependent upon the *keyword* that you supply.

The following table lists each keyword and the associated expressions:

Table 6-1 LDIF Target Keywords

Keyword	**Valid Expressions**	**Wildcard Allowed?**
`target`	ldap:///*distinguished_name*	yes
`targetattr`	*attribute*	yes
`targetfilter`	*LDAP_filter*	yes
`targattrfilters`	*LDAP_operation:LDAP_filter*	yes

In all cases, you must keep in mind that when you place an ACI on an entry, if it is not a leaf entry, the ACI also applies to all entries below it. For example, if you target the entry `ou=accounting,dc=example,dc=com`, the permissions you set will apply to all entries in the accounting branch of the `example.com` tree.

As a counter example, if you place an ACI on the `ou=accounting,dc=example,dc=com` entry, you cannot target the `uid=sarette,ou=people,dc=example,dc=com` entry because it is not located under the accounting tree.

Be wary of using `!=` when specifying an attribute you want to deny. ACLs are logically ORed, which means that if you created two ACLs

```
acl1: ( target=...)( targetattr!=a )(version 3.0; acl
"name";allow (...)..
```

```
acl2: ( target=...)( targetattr!=b )(version 3.0; acl
"name";allow (...)..
```

the result would be to allow all values of the target attribute. The first ACL (`acl1`) will allow `b` and the second ACL (`acl2`) will allow `a`. The result of these two ACLs will be the same as the one resulting from using an ACL of the form:

```
acl3: ( targetattr="*" ) allow (...) ...
```

Notice that nothing is denied. This could give rise to security problems.

When you want to deny access to a particular attribute, use `deny` in the permissions clause rather than using `allow` with `( targetattr != value )`. For example, usages such as these are recommended:

```
acl1: ( target=...)( targetattr=a )(version 3.0; acl "name";deny
(...)..
```

```
acl2: ( target=...)( targetattr=b )(version 3.0; acl "name";deny
(...)..
```

Targeting a Directory Entry

To target a directory entry (and the entries below it), you must use the `target` keyword.

The `target` keyword can accept a value of the following format:

`target="ldap:///`*distinguished_name*`"`

This identifies the distinguished name of the entry to which the access control rule applies. For example:

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

NOTE If the DN of the entry to which the access control rule applies contains a comma, you must escape the comma with a single backslash (\). For example:

```
(target="ldap:///uid=lfuentes,dc=example.com
Bolivia\,S.A.")
```

You can also use a wildcard when targeting a distinguished name using the `target` keyword. The wildcard indicates that any character or string or substring is a match for the wildcard. Pattern matching is based on any other strings that have been specified with the wildcard.

The following are legal examples of wildcard usage:

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

  Matches every entry in the entire `example.com` tree that has the `uid` attribute in the entry's RDN.

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

  Matches every entry directly under the `example.com` node with a `uid` ending in Anderson.

- `(target="ldap:///uid=C*A,dc=example,dc=com")`

  Matches every entry directly under the `example.com` node with a `uid` beginning with C and ending with A.

Depending on the position of the wildcard, it can apply to the full DN, not only to attribute values. Therefore, the wildcard can be used as a substitute for portions of the DN. For example, `uid=andy*,dc=example,dc=com` targets all the directory entries in the entire `example.com` tree with a matching uid attribute and not just the entries that are immediately below the `dc=example,dc=com` node. In other words, this target matches with longer expressions such as `uid=andy,ou=eng,dc=example,dc=com`, or `uid=andy,ou=marketing,dc=example,dc=com`.

Some other valid examples follow:

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

  Matches every entry in the entire `example.com` tree that has the `uid` attribute in the entry's RDN.

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

  Matches every entry in the `example.com` tree whose distinguished name contains the `uid` and `ou` attributes. Thus:

  ```
  uid=fchen,ou=Engineering,dc=example,dc=com
  ```

  or

  ```
  uid=claire,ou=Engineering,ou=people,dc=example,dc=com
  ```

  would match, but the following would not:

```
uid=bjensen,dc=example,dc=com
ou=Engineering,dc=example,dc=com
```

NOTE You cannot use wildcards in the suffix part of a distinguished name. That is, if your directory uses the suffixes `c=US` and `c=GB`, then you *cannot* use the following target to reference both suffixes:

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

Neither can you use a target such as `uid=bjensen,dc=*.com`.

Targeting Attributes

In addition to targeting directory entries, you can also target one or more attributes included in the targeted entries. This is useful when you want to deny or allow access to partial information about an entry. For example, you could allow access to only the common name, surname, and telephone number attributes of a given entry. Or you could deny access to sensitive information such as passwords.

You can specify that the target is equal or is not equal to a specific attribute. The attributes you supply do not need to be defined in the schema. This absence of schema checking makes it possible to implement an access control policy when you set up your directory service for the first time, even if the ACLs you create do not apply to the current directory content.

To target attributes, you use the `targetattr` keyword. The keyword uses the following syntax:

```
(targetattr = "attribute")
```

You can target multiple attributes by using the `targetattr` keyword with the following syntax:

```
(targetattr = "attribute1 || attribute2 ... || attributen")
```

Where *attribute* is the name of the attribute you want to target.

For example, to target the common name attribute you would use:

```
(targetattr = "cn")
```

To target an entry's common name, surname, and uid attributes, you would use the following:

```
(targetattr = "cn || sn || uid")
```

The attributes specified in the `targetattr` keyword apply to the entry that the ACI is targeting and to all the entries below it. If you target the password attribute on the entry `uid=bjensen,ou=Marketing,dc=example,dc=com`, only the password attribute on the `bjensen` entry is affected by the ACI because it is a leaf entry.

If, however, you target the tree's branch point `ou=Marketing,dc=example,dc=com`, then all the entries beneath the branch point that can contain a password attribute are affected by the ACI.

Targeting Both an Entry and Attributes

By default, the entry targeted by an ACI containing a `targetattr` keyword is the entry on which the ACI is placed. That is, if you put the ACI

```
aci: (targetattr = "uid")(access_control_rules;)
```

on the `ou=Marketing, dc=example,dc=com` entry, then the ACI applies to the entire Marketing subtree. However, you can also explicitly specify a target using the `target` keyword as follows:

```
aci: (target="ldap:///ou=Marketing,
dc=example,dc=com")(targetattr="uid") (access_control_rules;)
```

The order in which you specify the `target` and the `targetattr` keywords is not important.

Targeting Entries or Attributes Using LDAP Filters

You can use LDAP filters to target a group of entries that match certain criteria. To do this, you must use the `targetfilter` keyword with an LDAP filter.

The syntax of the `targetfilter` keyword is:

```
(targetfilter = "LDAP_filter")
```

where *LDAP_filter* is a standard LDAP search filter. For more information on the syntax of LDAP search filters, see Appendix B, "Finding Directory Entries."

For example, suppose that all entries in the accounting department include the attribute-value pair `ou=accounting`, and all entries in the engineering department include the attribute-value pair `ou=engineering` subtree. To target all the entries in the accounting and engineering branches of the directory tree, you could use the following filter:

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

This type of filter targets whole entries. You can associate the `targetfilter` and the `targetattr` keywords to create ACIs that apply to a subset of attributes in the targeted entries.

The following LDIF example allows members of the Engineering Admins group to modify the `departmentNumber` and `manager` attributes of all entries in the Engineering business category. This example uses LDAP filtering to select all entries with `businessCategory` attributes set to Engineering:

```
dn: dc=example,dc=com
objectClass: top
objectClass: organization
aci: (targetattr="departmentNumber || manager")
(targetfilter="(businessCategory=Engineering)")
(version 3.0; acl "eng-admins-write"; allow (write)
groupdn ="ldap:///cn=Engineering Admins, dc=example,dc=com";)
```

TIP Although using LDAP filters can be useful when you are targeting entries and attributes that are spread across the directory, the results are sometimes unpredictable because filters do not directly name the object for which you are managing access. The set of entries targeted by a filtered ACI is likely to change as attributes are added or deleted. Therefore, if you use LDAP filters in ACIs, you should verify that they target the correct entries and attributes by using the same filter in an `ldapsearch` operation.

Targeting Attribute Values Using LDAP Filters

You can use access control to target specific attribute values. This means that you can grant or deny permissions on an attribute if that attribute's value meets the criteria defined in the ACI. An ACI that grants or denies access based on an attribute's value is called a value-based ACI.

For example, you might grant all users in your organization permission to modify the `nsRoleDN` attribute in their own entry. However, you would also want to ensure that they do not give themselves certain key roles, such as "Top Level Administrator." LDAP filters are used to check that the conditions on attribute values are satisfied.

To create a value-based ACI, you must use the `targattrfilters` keyword with the following syntax:

```
(targattrfilters="add=attr1:F1 && attr2:F2... && attrn:Fn,del=attr1:F1
&& attr2:F2 ... && attrn:Fn")
```

where:

- `add` represents the operation of creating an attribute.
- `del` represents the operation of deleting an attribute.
- *attrx* represents the target attributes.
- *Fx* represents filters that apply only to the associated attribute.

When creating an entry, if a filter applies to an attribute in the new entry, then each instance of that attribute must satisfy the filter. When deleting an entry, if a filter applies to an attribute in the entry, then each instance of that attribute must also satisfy the filter.

When modifying an entry, if the operation adds an attribute, then the add filter that applies to that attribute must be satisfied; if the operation deletes an attribute, then the delete filter that applies to that attribute must be satisfied. If individual values of an attribute already present in the entry are replaced, then both the add and delete filters must be satisfied.

For example consider the following attribute filter:

```
(targattrfilters="add=nsroleDN:(!(nsRoleDN=cn=superAdmin)) &&
telephoneNumber:(telephoneNumber=123*)")
```

This filter can be used to allow users to add any role (`nsRoleDN` attribute) to their own entry, except the `superAdmin` role. It also allows users to add a telephone number with a 123 prefix.

NOTE You cannot create value-based ACIs from the Server Console.

Targeting a Single Directory Entry

Targeting a single directory entry is not straightforward because it goes against the design philosophy of the access control mechanism. However, it can be done:

- By creating a bind rule that matches user input in the bind request with an attribute value stored in the targeted entry. For more details, see “Defining Access Based on Value Matching,” on page 230.
- By using the `targetattr` and `targetfilter` keywords.

You can use the `targetattr` keyword to specify an attribute that is only present in the entry you want to target, and not in any of the entries below your target. For example, if you want to target `ou=people,dc=example,dc=com`, and there aren't any organizational units (`ou`) defined below that node, you could specify an ACI that contains:

```
targetattr=ou
```

A safer method is to use the `targetfilter` keyword and to specify explicitly an attribute value that appears in the entry alone. For example, during the installation of the Directory Server, the following ACI is created:

```
aci: (targetattr="*")(targetfilter=(o=NetscapeRoot))(version
3.0; acl "Default anonymous access"; allow (read, search)
userdn="ldap:///anyone";)
```

This ACI can apply only to the `o=NetscapeRoot` entry.

The risk associated with these method is that your directory tree might change in the future, and you would have to remember to modify this ACI.

Defining Permissions

Permissions specify the type of access you are allowing or denying. You can either allow or deny permission to perform specific operations in the directory. The various operations that can be assigned are known as *rights*.

There are two parts to setting permissions:

- Allowing or denying access
- Assigning rights

Allowing or Denying Access

You can either explicitly allow or deny access permissions to your directory tree. For more guidelines on when to allow and when to deny access, refer to the *Red Hat Directory Server Deployment Guide*.

NOTE From the Server Console, you cannot explicitly deny access, only grant permissions.

Assigning Rights

Rights detail the specific operations a user can perform on directory data. You can allow or deny all rights, or you can assign one or more of the following rights:

- **Read** — Indicates whether users can read directory data. This permission applies only to the search operation.
- **Write** — Indicates whether users can modify an entry by adding, modifying, or deleting *attributes*. This permission applies to the modify and modrdn operations.
- **Add** — Indicates whether users can create an *entry*. This permission applies only to the add operation.
- **Delete** — Indicates whether users can delete an *entry*. This permission applies only to the delete operation.
- **Search** — Indicates whether users can search for the directory data. Users must have Search and Read rights in order to view the data returned as part of a search result. This permission applies only to the search operation.
- **Compare** — Indicates whether the users can compare data they supply with data stored in the directory. With compare rights, the directory returns a success or failure message in response to an inquiry, but the user cannot see the value of the entry or attribute. This permission applies only to the compare operation.
- **Selfwrite** — Indicates whether users can add or delete their own DN from a group. This right is used only for group management.
- **Proxy** — Indicates whether the specified DN can access the target with the rights of another entry. For an overview of proxy access, refer to the *Red Hat Directory Server Deployment Guide*.
- **All** — Indicates that the specified DN has all rights (`read`, `write`, `search`, `delete`, `compare`, and `selfwrite`) to the targeted entry, *excluding* proxy rights.

Rights are granted independently of one another. This means, for example, that a user who is granted add rights can create an entry but cannot delete it if delete rights have not been specifically granted. Therefore, when planning the access control policy for your directory, you must ensure that you grant rights in a way that makes sense for users. For example, it doesn't usually make sense to grant write permission without granting read and search permissions.

NOTE The proxy mechanism is very powerful and must be used sparingly. Proxy rights are granted within the scope of the ACL, and there is no way to restrict who an entry that has the proxy right can impersonate—that is, when you grant a user proxy rights, that user has the ability to proxy for any user under the target; there is no way to restrict the proxy rights to only certain users. For example, if an entity has proxy rights to the `dc=example,dc=com` tree, that entity can do anything. So, make sure you set the proxy ACI at the lowest possible level of the DIT; see "Proxied Authorization ACI Example," on page 266.

For a general overview, see "Proxy Authentication" in chapter 7, "Designing a Secure Directory,"in the *Red Hat Directory Server Deployment Guide*.

Rights Required for LDAP Operations

This section describes the rights you need to grant to users depending on the type of LDAP operation you want to authorize them to perform.

- **Adding an entry:**
  - Grant add permission on the entry being added.
  - Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- **Deleting an entry:**
  - Grant delete permission on the entry to be deleted.
  - Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- **Modifying an attribute in an entry:**
  - Grant write permission on the attribute type.
  - Grant write permission on the value of each attribute type. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- **Modifying the RDN of an entry:**
  - Grant write permission on the entry.

- Grant write permission on the attribute type used in the new RDN.
- Grant write permission on the attribute type used in the old RDN, if you want to grant the right to delete the old RDN.
- Grant write permission on the value of attribute type used in the new RDN. This right is granted by default but could be restricted using the `targattrfilters` keyword.

- **Comparing the value of an attribute:**
  - Grant compare permission on the attribute type.
- **Searching for entries:**
  - Grant search permission on each attribute type used in the search filter.
  - Grant read permission on attribute types used in the entry.

The permissions you need to set up to allow users to search the directory are more readily understood with an example. Consider the following `ldapsearch` operation:

```
% ldapsearch -h host -s base -b "uid=bkolics,dc=example,dc=com"
objectclass=* mail
```

The following ACI is used to determine whether user `bkolics` can be granted access:

```
aci: (targetattr = "mail")(version 3.0; acl "self access to
mail"; allow (read, search) userdn = "ldap:///self";)
```

The search result list is empty because this ACI does not grant access to the `objectclass` attribute. If you want the search operation described above to be successful, you must modify the ACI to read as follows:

```
aci: (targetattr = "mail || objectclass")(version 3.0; acl "self
access to mail"; allow (read, search) userdn = "ldap:///self";)
```

Permissions Syntax

In an ACI statement, the syntax for permissions is:

```
allow|deny (rights)
```

where *rights* is a list of 1 to 8 comma-separated keywords enclosed within parentheses. Valid keywords are `read`, `write`, `add`, `delete`, `search`, `compare`, `selfwrite`, `proxy`, or `all`.

In the following example, read, search, and compare access is allowed, provided the bind rule is evaluated to be true:

```
aci:  (target="ldap:///dc=example,dc=com") (version 3.0;acl
"example";
allow (read, search, compare) bind_rule;)
```

Access Control and the modrdn Operation

To explicitly deny `modrdn` rights using ACIs, you must target the relevant entries but omit the `targetattr` keyword. For example, to prevent the `cn=helpDeskGroup,ou=groups,o=example.com` group from renaming any entries in the set specified by the pattern `cn=*,ou=people,o=example.com`, you would add the following ACI:

```
aci: (target="ldap:///cn=*,ou=people,o=example.com")
(version 3.0; acl "Deny modrdn rights to the helpDeskGroup";
deny(write)
groupdn="ldap:///cn=helpDeskGroup,ou=groups,o=example.com";)
```

Bind Rules

Depending on the ACIs defined for the directory, for certain operations, you need to *bind* to the directory. *Binding* means logging in or authenticating yourself to the directory by providing a bind DN and password, or, if using SSL, a certificate. The credentials provided in the bind operation and the circumstances of the bind determine whether access to the directory is allowed or denied.

Every permission set in an ACI has a corresponding bind rule that details the required credentials and bind parameters.

Bind rules can be simple. For example, a bind rule can simply state that the person accessing the directory must belong to a specific group. Bind rules can also be more complex. For example, a bind rule can state that a person must belong to a specific group and must log in from a machine with a specific IP address, between 8 a.m. and 5 p.m.

Bind rules define who can access the directory, when, and from where. More specifically, bind rules can specify:

- Users, groups, and roles that are granted access.
- Location from which an entity must bind.
- Time or day on which binding must occur.
- Type of authentication that must be in use during binding.

Additionally, bind rules can be complex constructions that combine these criteria by using Boolean operators. See "Using Boolean Bind Rules," on page 241, for more information.

Bind Rule Syntax

Whether access is allowed or denied depends on whether an ACI's bind rule is evaluated to be true. Bind rules use one of the two following patterns:

```
keyword = "expression";
keyword != "expression";
```

where equal (=) indicates that *keyword* and *expression* must match in order for the bind rule to be true, and not equal (!=) indicates that *keyword* and *expression* must not match in order for the bind rule to be true.

NOTE The timeofday keyword also supports the inequality expressions (<, <=, >, >=). This is the only keyword that supports these expressions.

The quotation marks (`""`) around *expression* and the delimiting semicolon (;) are required. The expressions you can use depend on the associated *keyword*.

The following table lists each keyword and the associated expressions. It also indicates whether wildcard characters are allowed in the expression.

Table 6-2 LDIF Bind Rule Keywords

Keyword	Valid Expressions	Wildcard Allowed?
`userdn`	ldap:///*distinguished_name*	yes, in DN only
	ldap:///all	
	ldap:///anyone	
	ldap:///self	
	ldap:///parent	
	ldap:///*suffix*??sub?(*filter*)	
`groupdn`	ldap:///*DN* || *DN*	no
`roledn`	ldap:///*DN* || *DN*	no

Table 6-2 LDIF Bind Rule Keywords *(Continued)*

Keyword	Valid Expressions	Wildcard Allowed?
`userattr`	*attribute#bindType* or *attribute#value*	no
`ip`	*IP_address*	yes
`dns`	*DNS_host_name*	yes

`dayofweek`	`sun`
	`mon`
	`tue`
	`wed`
	`thu`
	`fri`
	`sat`
		no

`timeofday`	`0 - 2359`	no

`authmethod`	none
	`simple`
	`ssl`
	`sasl` *authentication_method*
		no

The sections that follow contain further detail on bind rule syntax for each keyword.

Defining User Access - userdn Keyword

User access is defined using the `userdn` keyword. The `userdn` keyword requires one or more valid distinguished names in the following format :

```
userdn = "ldap:///dn [|| ldap:///dn]...[||ldap:///dn]"
```

where *dn* can be a DN or one of the expressions `anyone`, `all`, `self`, or `parent`:

```
userdn = "ldap:///anyone" - defines anonymous access

userdn = "ldap:///all" - defines general access

userdn = "ldap:///self" - defines self access

userdn = "ldap:///parent" - defines access for the parent entry
```

The `userdn` keyword can also be expressed as an LDAP filter of the form:

```
ldap:///suffix??sub?(filter)
```

NOTE If a DN contains a comma, the comma must be preceded by a backslash (\) escape character.

Anonymous Access (anyone Keyword)

Granting anonymous access to the directory means that anyone can access it without providing a bind DN or password and regardless of the circumstances of the bind. You can limit anonymous access to specific types of access (for example, access for read or access for search) or to specific subtrees or individual entries within the directory.

From the Server Console, you define anonymous access through the Access Control Editor. See "Creating ACIs from the Console," on page 242.

General Access (all Keyword)

You can use bind rules to indicate that a permission applies to anyone who has successfully bound to the directory; that is, all authenticated users. This allows general access while preventing anonymous access.

From the Server Console, you define general access on the Access Control Editor. For more information, see "Creating ACIs from the Console," on page 242.

Self Access (self Keyword)

Specifies that users are granted or denied access to their own entries. In this case, access is granted or denied if the bind DN matches the DN of the targeted entry.

From the Server Console, you set up self access on the Access Control Editor. For more information, see "Creating ACIs from the Console," on page 242.

Parent Access (parent Keyword)

Specifies that users are granted or denied access to the entry only if their bind DN is the parent of the targeted entry.

You cannot set up parent access control using the Server Console.

LDAP URLs

You can dynamically target users in ACIs using a URL with a filter as follows:

```
userdn = "ldap:///<suffix>??sub?(filter)"
```

For example, all users in the accounting and engineering branches of the `example.com` tree would be granted or denied access to the targeted resource dynamically based on the following URL:

```
userdn = "ldap:///dc=example,dc=com??sub?(|(ou=engineering)
(ou=accounting))"
```

NOTE	Do not specify a hostname or port number within the LDAP URL. LDAP URLs always apply to the local server.

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

Wildcards

You can also specify a set of users by using the wildcard character (*). For example, specifying a user DN of `uid=u*,dc=example,dc=com` indicates that only users with a bind DN beginning with the letter `u` will be allowed or denied access based on the permissions you set.

From the Server Console, you set user access from the Access Control Editor. For more information, see "Creating ACIs from the Console," on page 242.

Examples

This section contains examples of the `userdn` syntax.

- **Userdn keyword containing an LDAP URL:**

  ```
  userdn = "ldap:///uid=*,dc=example,dc=com";
  ```

  The bind rule is evaluated to be true if the user binds to the directory using any distinguished name of the specified pattern. For example, both of the following bind DNs would be evaluated to be true:

  ```
  uid=ssarette,dc=example,dc=com
  uid=tjaz,ou=Accounting,dc=example,dc=com
  ```

  whereas the following bind DN would be evaluated to be false:

  ```
  cn=Babs Jensen,dc=example,dc=com
  ```

- **Userdn keyword containing logical OR of LDAP URLs:**

  ```
  userdn="ldap:///uid=bj,c=example.com ||
  ldap:///uid=kc,dc=example,dc=com";
  ```

  The bind rule is evaluated to be true if the client binds as either of the two supplied distinguished names.

- **Userdn keyword excluding a specific LDAP URL:**

  ```
  userdn != "ldap:///uid=*,ou=Accounting,dc=example,dc=com";
  ```

  The bind rule is evaluated to be true if the client is not binding as a UID-based distinguished name in the accounting subtree. This bind rule only makes sense if the targeted entry is not under the accounting branch of the directory tree.

- **Userdn keyword containing self keyword:**

  ```
  userdn = "ldap:///self";
  ```

  The bind rule is evaluated to be true if the user is accessing the entry represented by the DN with which the user bound to the directory. That is, if the user has bound as `uid=ssarette, dc=example,dc=com` and the user is attempting an operation on the `uid=ssarette,dc=example,dc=com` entry, then the bind rule is true.

  If you want to grant all users in the `example.com` tree write access to their `userPassword` attribute, you would create the following ACI on the `dc=example,dc=com` node.

  ```
  aci: (targetattr = "userPassword") (version 3.0; acl
  "write-self"; allow (write) userdn = "ldap:///self";)
  ```

- **Userdn keyword containing the all keyword:**

  ```
  userdn = "ldap:///all";
  ```

  The bind rule is evaluated to be true for any valid bind DN. To be true, a valid distinguished name and password must have been presented by the user during the bind operation.

  For example, if you want to grant read access to the entire tree to all authenticated users, you would create the following ACI on the `dc=example,dc=com` node:

  ```
  aci:(version 3.0; acl "all-read"; allow (read)
  userdn="ldap:///all";)
  ```

- **Userdn keyword containing the anyone keyword:**

  ```
  userdn = "ldap:///anyone";
  ```

  The bind rule is evaluated to be true for anyone; use this keyword to provide anonymous access to your directory.

  For example, if you want to allow anonymous read and search access to the entire `example.com` tree, you would create the following ACI on the `dc=example,dc=com` node:

```
aci: (version 3.0; acl "anonymous-read-search"; allow (read,
search) userdn = "ldap:///anyone";)
```

- **Userdn keyword containing the parent keyword:**

  ```
  userdn = "ldap:///parent";
  ```

  The bind rule is evaluated to be true if the bind DN is the parent of the targeted entry.

  For example, if you want to grant write access to every user's child entries, you would create the following ACI on the `dc=example,dc=com` node:

  ```
  aci:(version 3.0; acl "parent access"; allow (write)
  userdn="ldap:///parent";)
  ```

  ```
  userdn = "ldap:///dc=example,dc=com???(|(ou=engineering)
  (ou=sales))";
  ```

  The bind rule is evaluated to be true if the user belongs to the engineering or sales subtree.

Defining Group Access - groupdn Keyword

Members of a specific group can access a targeted resource. This is known as *group access*. Group access is defined using the `groupdn` keyword to specify that access to a targeted entry will be granted or denied if the user binds using a DN that belongs to a specific group.

The `groupdn` keyword requires one or more valid distinguished names in the following format :

```
groupdn="ldap:///dn [|| ldap:///dn]...[|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the named group.

NOTE If a DN contains a comma, the comma must be escaped by a backslash (\).

From the Server Console, you can define specific groups using the Access Control Editor. For more information, see "Creating ACIs from the Console," on page 242.

Examples

This section contains examples of the `groupdn` syntax.

- **Groupdn keyword containing an LDAP URL:**

  ```
  groupdn = "ldap:///cn=Administrators,dc=example,dc=com";
  ```

  The bind rule is evaluated to be true if the bind DN belongs to the Administrators group. If you wanted to grant the Administrators group permission to write to the entire directory tree, you would create the following ACI on the `dc=example,dc=com` node:

  ```
  aci: (version 3.0; acl "Administrators-write"; allow (write)
  groupdn="ldap:///cn=Administrators,dc=example,dc=com";)
  ```

- **Groupdn keyword containing logical OR of LDAP URLs:**

  ```
  groupdn = "ldap:///cn=Administrators,dc=example,dc=com" ||
  "ldap:///cn=Mail Administrators,dc=example,dc=com";
  ```

  The bind rule is evaluated to be true if the bind DN belongs to either the Administrators or the Mail Administrators group.

Defining Role Access - roledn Keyword

Members of a specific role can access a targeted resource. This is known as *role access*. Role access is defined using the `roledn` keyword to specify that access to a targeted entry will be granted or denied if the user binds using a DN that belongs to a specific role.

The `roledn` keyword requires one or more valid distinguished names in the following format :

```
roledn = "ldap:///dn [|| ldap:///dn]... [|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the specified role.

NOTE If a DN contains a comma, the comma must be escaped by a backslash (\).

The `roledn` keyword has the same syntax and is used in the same way as the `groupdn` keyword.

Defining Access Based on Value Matching

You can set bind rules to specify that an attribute value of the entry used to bind to the directory must match an attribute value of the targeted entry.

For example, you can specify that the bind DN must match the DN in the `manager` attribute of a user entry in order for the ACI to apply. In this case, only the user's manager would have access to the entry.

This example is based on DN matching. However, you can match any attribute of the entry used in the bind with the targeted entry. For example, you could create an ACI that allowed any user whose `favoriteDrink` attribute is `beer` to read all the entries of other users that have the same value for `favoriteDrink`.

Using the userattr Keyword

The `userattr` keyword can be used to specify which attribute values must match between the entry used to bind and the targeted entry. You can specify:

- A user DN
- A group DN
- A role DN
- An LDAP filter, in an LDAP URL
- Any attribute type

The LDIF syntax of the `userattr` keyword is as follows:

```
useratr = "attrName#bindType"
```

or, if you are using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter:

```
userattr = "attrName#attrValue"
```

where:

- *attrName* is the name of the attribute used for value matching.
- *bindType* is one of `USERDN`, `GROUPDN`, or `LDAPURL`.
- *attrValue* is any string representing an attribute value.

The following sections provide examples of the `userattr` keyword with the various possible bind types.

Example with USERDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on the user DN:

```
userattr = "manager#USERDN"
```

The bind rule is evaluated to be true if the bind DN matches the value of the `manager` attribute in the targeted entry. You can use this to allow a user's manager to modify employees' attributes. This mechanism only works if the `manager` attribute in the targeted entry is expressed as a full DN.

The following example grants a manager full access to his or her employees' entries:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)
(version 3.0;
acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

Example with GROUPDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on a group DN:

```
userattr = "owner#GROUPDN"
```

The bind rule is evaluated to be true if the bind DN is a member of the group specified in the `owner` attribute of the targeted entry. For example, you can use this mechanism to allow a group to manage employees' status information. You can use an attribute other than `owner` as long as the attribute you use contains the DN of a group entry.

The group you point to can be a dynamic group, and the DN of the group can be under any suffix in the database. However, the evaluation of this type of ACI by the server is very resource intensive.

If you are using static groups that are under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?owner#GROUPDN"
```

In this example, the group entry is under the `dc=example,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

(By default, `owner` is not an allowed entry in a user's entry. You would have to extend your schema to allow this attribute in a `person` object.)

Example with ROLEDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on a role DN:

```
userattr = "exampleEmployeeReportsTo#ROLEDN"
```

The bind rule is evaluated to be true if the bind DN belongs to the role specified in the `exampleEmployeeReportsTo` attribute of the targeted entry. For example, if you create a nested role for all managers in your company, you can use this mechanism to grant managers at all levels access to information about employees that are at a lower grade than themselves.

NOTE This example assumes that you have added the `exampleEmployeeReportsTo` attribute to the schema and that all employee entries contain this attribute. It also assumes that the value of this attribute is the DN of a role entry.

For information on designing your schema, refer to *Red Hat Directory Server Deployment Guide*. For information on adding attributes to the schema, see "Creating Attributes," on page 383.

The DN of the role can be under any suffix in the database. If, in addition, you are using filtered roles, the evaluation of this type of ACI uses a lot of resources on the server.

If you are using a static role definition and the role entry is under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?employeeReportsTo#ROLEDN"
```

In this example, the role entry is under the `dc=example,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

Example with LDAPURL Bind Type

The following is an example of the `userattr` keyword associated with a bind based on an LDAP filter:

`userattr = "`*myfilter#LDAPURL*`"`

The bind rule is evaluated to be true if the bind DN matches the filter specified in the *myfilter* attribute of the targeted entry. The *myfilter* attribute can be replaced by any attribute that contains an LDAP filter.

Example with Any Attribute Value

The following is an example of the `userattr` keyword associated with a bind based on any attribute value:

```
userattr = "favoriteDrink#Beer"
```

The bind rule is evaluated to be true if the bind DN and the target DN include the `favoriteDrink` attribute with a value of `Beer`.

Using the userattr Keyword with Inheritance

When you use the `userattr` keyword to associate the entry used to bind with the target entry, the ACI applies only to the target specified and not to the entries below it. In some circumstances, you might want to extend the application of the ACI several levels below the targeted entry. This is possible by using the parent keyword and specifying the number of levels below the target that should inherit the ACI.

When you use the `userattr` keyword in association with the `parent` keyword, the syntax is as follows:

```
userattr = "parent[inheritance_level].attrName#bindType"
```

or, if you are using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter:

```
userattr = "parent[inheritance_level].attrName#attrValue"
```

where:

- *inheritance_level* is a comma-separated list that indicates how many levels below the target will inherit the ACI. You can include five levels `[0,1,2,3,4]` below the targeted entry; zero (`0`) indicates the targeted entry.
- *attribute* is the attribute targeted by the `userattr` or `groupattr` keyword.
- *bindType* can be one of `USERDN`, `GROUPDN`, or `LDAPURL`.

For example,

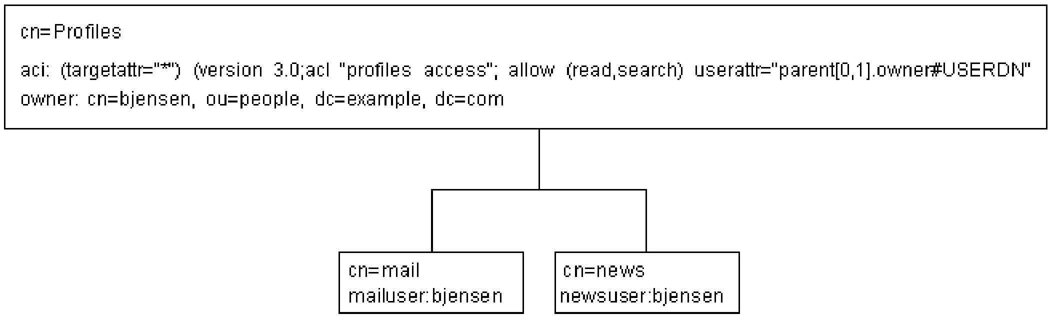
```
userattr = "parent[0,1].manager#USERDN"
```

This bind rule is evaluated to be true if the bindDN matches the manager attribute of the targeted entry. The permissions granted when the bind rule is evaluated to be true apply to the target entry *and* to all entries immediately below it.

Example with userattr Inheritance

The example in Figure 6-1 indicates that user `bjensen` is allowed to read and search the `cn=Profiles` entry as well as the first level of child entries which includes `cn=mail` and `cn=news`, thus allowing her to search through her own mail and news IDs.

Figure 6-1 Using Inheritance With the userattr Keyword


cn=Profiles
aci: (targetattr="*") (version 3.0;acl "profiles access"; allow (read,search) userattr="parent[0,1].owner#USERDN"
owner: cn=bjensen, ou=people, dc=example, dc=com
cn=mail
mailuser:bjensen
cn=news
newsuser:bjensen

In this example, if you did not use inheritance, you would have to do one of the following to achieve the same result:

- Explicitly set read and search access for user `bjensen` on the `cn=Profiles`, `cn=mail`, and `cn=news` entries in the directory.
- Add the owner attribute with a value of `bjensen` to the `cn=mail` and `cn=news` entries, and then add the following ACI to the `cn=mail` and `cn=news` entries.

```
aci: (targetattr="*") (version 3.0; acl "profiles access";
allow (read,search) userattr="owner#USERDN";)
```

Granting Add Permission Using the userattr Keyword

If you use the `userattr` keyword in conjunction with `all` or `add` permissions, you might find that the behavior of the server is not what you expect. Typically, when a new entry is created in the directory, Directory Server evaluates access rights on the entry being created and not on the parent entry. However, in the case of ACIs using the `userattr` keyword, this behavior could create a security hole, and the server's normal behavior is modified to avoid it.

Consider the following example:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*) (version
3.0;
acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

This ACI grants managers all rights on the entries of employees that report to them. However, because access rights are evaluated on the entry being created, this type of ACI would also allow any employee to create an entry in which the manager attribute is set to their own DN. For example, disgruntled employee Joe (`cn=Joe,ou=eng,dc=example,dc=com`) might want to create an entry in the Human Resources branch of the tree to use (or misuse) the privileges granted to Human Resources employees.

He could do this by creating the following entry:

```
dn: cn= Trojan Horse,ou=Human Resources,dc=example,dc=com
objectclass: top
...
cn: Trojan Horse
manager: cn=Joe,ou=eng,dc=example,dc=com
```

To avoid this type of security threat, the ACI evaluation process does not grant add permission at level 0, to the entry itself. You can, however, use the `parent` keyword to grant add rights below existing entries. You must specify the number of levels below the parent for add rights. For example, the following ACI allows child entries to be added to any entry in the `dc=example,dc=com` that has a `manager` attribute that matches the bind DN:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)
(version 3.0; acl "parent-access"; allow (add)
userattr = "parent[0,1].manager#USERDN";)
```

This ACI ensures that add permission is granted only to users whose bind DN matches the manager attribute of the parent entry.

Defining Access from a Specific IP Address

Using bind rules, you can indicate that the bind operation must originate from a specific IP address. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on an IP address is as follows:

`ip = "`*IP_address*`" or ip != "`*IP_address*`"`

The IP address must be expressed in dot notation.You can use the wildcard character (*) to include multiple machines. For example, the following string is valid:

```
ip = "12.123.1.*";
```

The bind rule is evaluated to be true if the client accessing the directory is located at the named IP address. This can be useful for allowing certain kinds of directory access only from a specific subnet or machine.

For example, you could use a wildcard IP address such as `12.3.45.*` to specify a specific subnetwork or `123.45.6.*+255.255.255.115` to specify a subnetwork mask.

From the Server Console, you can define specific machines to which the ACI applies through the Access Control Editor. For more information, see "Creating ACIs from the Console," on page 242.

Defining Access from a Specific Domain

A bind rule can specify that the bind operation must originate from a particular domain or host machine. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on the DNS hostname is as follows:

```
dns = "DNS_Hostname" or dns != "DNS_Hostname"
```

CAUTION The `dns` keyword requires that the naming service used on your machine is DNS. If the name service is not DNS, you should use the `ip` keyword instead.

The `dns` keyword requires a fully qualified DNS domain name. Granting access to a host without specifying the domain creates a potential security threat. For example, the following expression is allowed but not recommended:

```
dns = "legend.eng";
```

You should use a fully qualified name such as:

```
dns = "legend.eng.example.com";
```

The dns keyword allows wildcards. For example:

```
dns = "*.example.com";
```

The bind rule is evaluated to be true if the client accessing the directory is located in the named domain. This can be useful for allowing access only from a specific domain. Wildcards will not work if your system uses a naming service other than DNS. In such a case, if you want to restrict access to a particular domain, use the ip keyword, as described in "Defining Access from a Specific IP Address," on page 236.

Defining Access at a Specific Time of Day or Day of Week

You can use bind rules to specify that binding can only occur at a certain time of day or on a certain day of the week. For example, you can set a rule that will allow access only if it is between the hours of 8 a.m. and 5 p.m. Monday through Friday. The time used to evaluate access rights is the time on the Directory Server, not the time on the client.

The LDIF syntax for setting a bind rule based on the time of day is as follows:

```
timeofday operator "time"
```

where *operator* can be one of the following symbols: equal to (=), not equal to (!=), greater than (>), greater than or equal to (>=), less than (<), or less than or equal to (<=).

The `timeofday` keyword requires a time of day expressed in hours and minutes in the 24 hour clock (0 to 2359).

NOTE The time on the server is used for the evaluation, not the time on the client.

The LDIF syntax for setting a bind rule based on the day in the week is as follows:

```
dayofweek = "day1, day2 ..."
```

The possible values for the `dayofweek` keyword are the English three-letter abbreviations for the days of the week: `sun`, `mon`, `tue`, `wed`, `thu`, `fri`, `sat`.

Examples

The following are examples of the `timeofday` and `dayofweek` syntax:

```
timeofday = "1200";
```

The bind rule is evaluated to be true if the client is accessing the directory at noon.

```
timeofday != "0100";
```

The bind rule is evaluated to be true if the client is accessing the directory at any time other than 1 a.m.

```
timeofday > "0800";
```

The bind rule is evaluated to be true if the client is accessing the directory at any time after 8 a.m.

```
timeofday < "1800";
```

The bind rule is evaluated to be true if the client is accessing the directory at any time before 6 p.m.

```
timeofday >= "0800";
```

The bind rule is evaluated to be true if the client is accessing the directory at 8 a.m. or later.

```
timeofday <= "1800";
```

The bind rule is evaluated to be true if the client is accessing the directory at 6 p.m. or earlier.

```
dayofweek = "Sun, Mon, Tue";
```

The bind rule is evaluated to be true if the client is accessing the directory on Sunday, Monday, or Tuesday.

Defining Access Based on Authentication Method

You can set bind rules that state that a client must bind to the directory using a specific authentication method. The authentication methods available are:

- None — Authentication is not required. This is the default. It represents anonymous access.
- Simple — The client must provide a user name and password to bind to the directory.
- SSL — The client must bind to the directory over a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection.

In the case of SSL, the connection is established to the LDAPS second port; in the case of TLS, the connection is established through a Start TLS operation. In both cases, a certificate must be provided. For information on setting up SSL, see chapter 11, "Managing SSL and SASL."

- SASL — The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. Directory Server supports three SASL mechanisms: EXTERNAL, DIGEST-MD5, and GSS-API for Kerberos systems. For information on setting up SASL, see chapter 11, "Managing SSL and SASL."

You cannot set up authentication-based bind rules through the Access Control Editor.

The LDIF syntax for setting a bind rule based on an authentication method is as follows:

`authmethod = "`*authentication_method*`"`

where *authentication_method* is `none`, `simple`, `ssl`, or `"sasl` *sasl_mechanism*`"`.

Examples

The following are examples of the `authmethod` keyword:

```
authmethod = "none";
```

Authentication is not checked during bind rule evaluation.

```
authmethod = "simple";
```

The bind rule is evaluated to be true if the client is accessing the directory using a username and password.

```
authmethod = "ssl";
```

The bind rule is evaluated to be true if the client authenticates to the directory using a certificate over LDAPS. This is not evaluated to be true if the client authenticates using simple authentication (bind DN and password) over LDAPS.

```
authmethod = "sasl DIGEST-MD5";
```

The bind rule is evaluated to be true if the client is accessing the directory using the SASL DIGEST-MD5 mechanism. The other supported SASL mechanisms are EXTERNAL and GSS-API.

Using Boolean Bind Rules

Bind rules can be complex expressions that use the Boolean expressions AND, OR, and NOT to set very precise access rules. You cannot use the Server Console to create Boolean bind rules. You must create an LDIF statement.

The LDIF syntax for a Boolean bind rule is as follows:

bind_rule [*boolean*][*bind_rule*][*boolean*][*bind_rule*]...;)

For example, the following bind rule will be evaluated to be true if the bind DN is a member of either the administrator's group or the mail administrator's group and if the client is running from within the example.com domain:

```
(groupdn = "ldap:///cn=administrators,dc=example,dc=com" or
groupdn = "ldap:///cn=mail administrators,dc=example,dc=com" and
dns = "*.example.com";)
```

The trailing semicolon (;) is a required delimiter that must appear after the final bind rule.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.
- NOT before AND or OR operators.

The Boolean OR and Boolean AND operators have no order of precedence.

Consider the following Boolean bind rules:

(*bind_rule_A*) OR (*bind_rule_B*)

(*bind_rule_B*) OR (*bind_rule_A*)

Because Boolean expressions are evaluated from left to right, in the first case, bind rule A is evaluated before bind rule B, and, in the second case, bind rule B is evaluated before bind rule A.

However, the Boolean NOT is evaluated *before* the Boolean OR and Boolean AND. Thus, in the following example

(*bind_rule_A*) AND NOT (*bind_rule_B*)

bind rule B is evaluated before bind rule A despite the left-to-right rule.

Creating ACIs from the Console

You can use the Directory Server Console to view, create, edit, and delete access control instructions for your directory. This section provides general instructions for:

- Displaying the Access Control Editor
- Viewing Current ACIs
- Creating a New ACI
- Editing an ACI
- Deleting an ACI

See "Access Control Usage Examples," on page 247, for a collection of access control rules commonly used in Directory Server security policies, along with step-by-step instructions for using the Directory Server Console to create them.

The Access Control Editor does not enable you to construct some of the more complex ACIs when you are in Visual editing mode. In particular, from the Access Control Editor you cannot:

- Deny access (see "Permissions Syntax," on page 222).
- Create value-based ACIs (see "Targeting Attribute Values Using LDAP Filters," on page 217).
- Define parent access (see "Parent Access (parent Keyword)," on page 226).
- Create ACIs that contain Boolean bind rules (see "Using Boolean Bind Rules," on page 241).
- Generally, create ACIs that use the following keywords: `roledn`, `userattr`, `authmethod`.

TIP In the Access Control Editor, you can click on the Edit Manually button at any time to check the LDIF representation of the changes you make through the graphical interface.

Displaying the Access Control Editor

1. Start the Directory Server Console. Log in using the bind DN and password of a privileged user, such as the Directory Manager, who has write access to the ACIs configured for the directory.

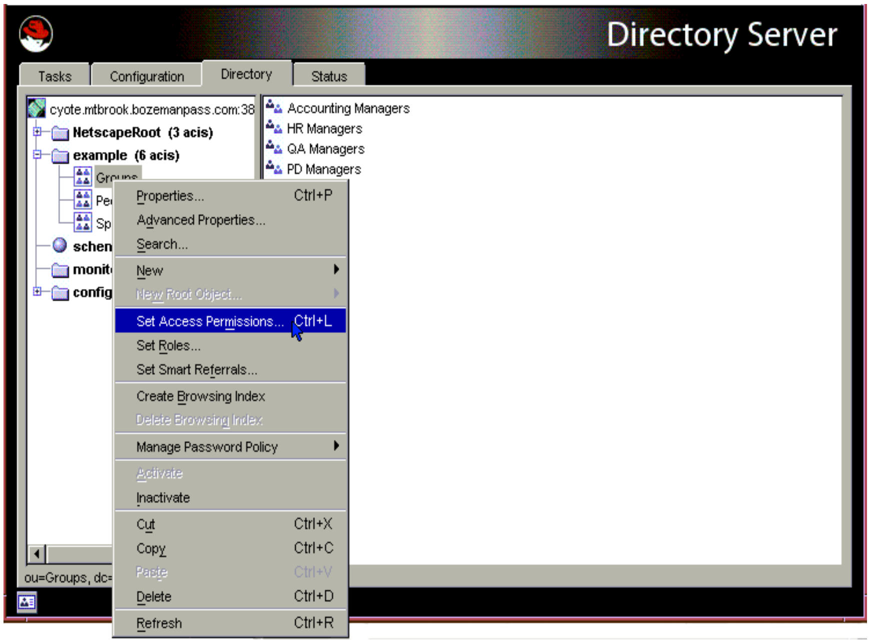
   For instructions, refer to “Using the Directory Server Console,” on page 34.

2. In the Directory Server Console, select the Directory tab.

3. Right-click the entry in the navigation tree for which you want to set access control, and select Set Access Permissions from the pop-up menu (Figure 6-2).

   Or highlight the entry, and select Set Access Permissions from the Object menu.

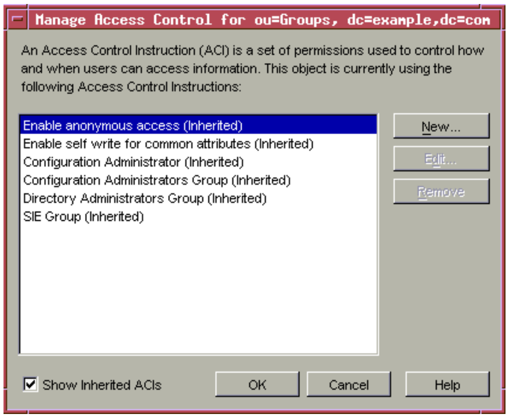
Figure 6-2 Selecting an Object in the Navigation Tree to Set Access Control



4. Click New.

   The Access Control Editor is displayed as shown in Figure 6-3.

Figure 6-3 Access Control Editor Window


Manage Access Control for ou=Groups, dc=example,dc=com

An Access Control Instruction (ACI) is a set of permissions used to control how and when users can access information. This object is currently using the following Access Control Instructions:

Enable anonymous access (Inherited)
Enable self write for common attributes (Inherited)
Configuration Administrator (Inherited)
Configuration Administrators Group (Inherited)
Directory Administrators Group (Inherited)
SIE Group (Inherited)

For information on navigating through the Access Control dialog boxes, refer to the online help.

Viewing Current ACIs

If you want to see what ACIs apply to a particular subtree in your directory, follow these steps:

1. In the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. Check the checkbox for Show Inherited ACIs if you want to display the full list of ACIs that apply to the entry.

Creating a New ACI

To create a new ACI:

1. Display the Access Control Editor.

   This task is explained in "Displaying the Access Control Editor," on page 243.

   If the view displayed is different from Figure 6-3, on page 244, click the Edit Visually button.

2. Name the ACI by typing a name in the ACI Name text box.

   The name can be any string you want to use to identify uniquely the ACI. If you do not enter a name, the server uses `unnamed ACI`.

3. In the Users/Groups tab, select the users to whom you are granting access by highlighting All Users or clicking the Add button to search the directory for the users to add.

   In the Add Users and Groups window:

   a. Select a search area from the drop-down list, enter a search string in the Search field, and click the Search button.

      The search results are displayed in the list below.

   b. Highlight the entries you want in the search result list, and click the Add button to add them to the list of entries which have access permission.

   c. Click OK to dismiss the Add Users and Groups window.

      The entries you selected are now listed on the Users/Groups tab in the ACI editor.

4. In the Access Control Editor, click the Rights tab, and use the checkboxes to select the rights to grant.

5. Click the Targets tab, then click This Entry to display the node targeted by the ACI.

   You can change the value of the target DN, but the new DN must be a direct or indirect child of the selected entry.

   If you do not want every entry in the subtree under this node to be targeted by the ACI, you must enter a filter in the Filter for Sub-entries field.

   Additionally, you can restrict the scope of the ACI to only certain attributes by selecting the attributes you want to target in the attribute list.

6. Click the Hosts tab, then the Add button, to display the Add Host Filter dialog box.

   You can specify a hostname or an IP address. If you specify an IP address, you can use the wildcard character (*).

7. Click the Times tab to display the table showing at what times access is allowed.

   By default, access is allowed at all times. You can change the access times by clicking and dragging the cursor over the table. You cannot select discrete blocks of time.

8. When you have finished editing the ACI, click OK.

   The ACI Editor is dismissed, and the new ACI is listed in the ACI Manager window.

NOTE At any time during the creation of the ACI, you can click the Edit Manually button to display the LDIF statement that corresponds to your input. You can modify this statement, but your changes will not necessarily be visible in the graphical interface.

Editing an ACI

To edit an ACI:

1. In the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. In the Access Control Manager window, highlight the ACI that you want to edit, and click Edit.

   The Access Control Editor is displayed. For details on the information you can edit using this dialog box, refer to the online help.

3. Make the changes you want under the various tabs of the Access Control Editor.

4. When you have finished editing the ACI, click OK.

   The ACI Editor is dismissed, and the modified ACI is listed in the ACI Manager.

Deleting an ACI

To delete an ACI:

1. In the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. In the Access Control Manager window, select the ACI that you want to delete.
3. Click Remove.

   The ACI is no longer listed in the Access Control Manager.

Access Control Usage Examples

The examples provided in this section illustrate how an imaginary ISP company, `example.com`, would implement its access control policy. All the examples explain how to perform a given task from the Console and using an LDIF file.

`example.com`'s business is to offer a web hosting service and Internet access. Part of `example.com`'s web hosting service is to host the directories of client companies. `example.com` actually hosts and partially manages the directories of two medium-sized companies, `HostedCompany1` and `HostedCompany2`. It also provides Internet access to a number of individual subscribers.

These are the access control rules that `example.com` wants to put in place:

- Grant anonymous access for read, search, and compare to the entire `example.com` tree for `example.com` employees (see "Granting Anonymous Access," on page 248).
- Grant write access to `example.com` employees for personal information, such as `homeTelephoneNumber` and `homeAddress` (see "Granting Write Access to Personal Entries," on page 251).
- Grant `example.com` employees the right to add any role to their entry, except certain critical roles (see "Restricting Access to Key Roles," on page 254).
- Grant the `example.com` Human Resources group all rights on the entries in the People branch (see "Granting a Group Full Access to a Suffix," on page 255).

- Grant all `example.com` employees the right to create group entries under the Social Committee branch of the directory and to delete group entries that they own (see “Granting Rights to Add and Delete Group Entries,” on page 257).
- Grant all `example.com` employees the right to add themselves to group entries under the Social Committee branch of the directory (see “Allowing Users to Add or Remove Themselves from a Group,” on page 264).
- Grant access to the directory administrator (role) of `HostedCompany1` and `HostedCompany2` on their respective branches of the directory tree, with certain conditions such as SSL authentication, time and date restrictions, and specified location (see “Granting Conditional Access to a Group or Role,” on page 259).
- Grant individual subscribers access to their own entries (see “Granting Write Access to Personal Entries,” on page 251).
- Deny individual subscribers access to the billing information in their own entries (see “Denying Access,” on page 261).
- Grant anonymous access to the world to the individual subscribers subtree, except for subscribers who have specifically requested to be unlisted. (This part of the directory could be a consumer server outside of the firewall and be updated once a day.) See “Granting Anonymous Access,” on page 248, and “Setting a Target Using Filtering,” on page 264.

Granting Anonymous Access

Most directories are run such that you can anonymously access at least one suffix for read, search, or compare. For example, you might want to set these permissions if you are running a corporate personnel directory that you want employees to be able to search, such as a phonebook. This is the case at `example.com` internally and is illustrated in the ACI “Anonymous example.com” example.

As an ISP, `example.com` also wants to advertise the contact information of all of its subscribers by creating a public phonebook accessible to the world. This is illustrated in the ACI “Anonymous World” example.

ACI “Anonymous example.com”

In LDIF, to grant read, search, and compare permissions to the entire `example.com` tree to `example.com` employees, you would write the following statement:

```
aci: (targetattr !="userPassword")(version 3.0; acl "Anonymous
Example"; allow (read, search, compare) userdn= "ldap:///anyone"
and dns="*.example.com";)
```

This example assumes that the `aci` is added to the `dc=example,dc=com` entry. The `userPassword` attribute is excluded from the scope of the ACI.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab in the ACI name field, type `Anonymous example.com`. Check that All Users is displayed in the list of users granted access permission.
4. In the Rights tab, tick the checkboxes for `read`, `compare`, and `search` rights. Make sure the other checkboxes are clear.
5. In the Targets tab, click This Entry to display the `dc=example,dc=com` suffix in the target directory entry field. In the attribute table, locate the `userPassword` attribute, and clear the corresponding checkbox.

   All other checkboxes should be ticked. This task is made easier if you click the Name header to organize the list of attributes alphabetically.
6. In the Hosts tab, click Add, and in the DNS host filter field, type `*.example.com`. Click OK to dismiss the dialog box.
7. Click OK in the Access Control Editor window.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI “Anonymous World”

In LDIF, to grant read and search access of the individual subscribers subtree to the world, while denying access to information on unlisted subscribers, you could write the following statement:

```
aci: (targetfilter= "(!(unlistedSubscriber=yes))")
(targetattr="homePostalAddress || homePhone || mail") (version
3.0; acl "Anonymous World"; allow (read, search) userdn=
"ldap:///anyone";)
```

This example assumes that the ACI is added to the `ou=subscribers,dc=example,dc=com` entry. It also assumes that every subscriber entry has an `unlistedSubscriber` attribute which is set to `yes` or `no`. The target definition filters out the unlisted subscribers based on the value of this attribute. For details on the filter definition, refer to "Setting a Target Using Filtering," on page 264.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the Subscribers entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Anonymous World`. Check that All Users is displayed in the list of users granted access permission.
4. In the Rights tab, tick the checkboxes for `read` and `search` rights. Make sure the other checkboxes are clear.
5. In the Targets tab, click This Entry to display the `dc=subscribers, dc=example,dc=com` suffix in the target directory entry field.
    a. In the filter for subentries field, type the following filter:

    ```
    (!(unlistedSubscriber=yes))
    ```

    b. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

    All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.
6. Click OK.

    The new ACI is added to the ones listed in the Access Control Manager window.

Granting Write Access to Personal Entries

Many directory administrators want to allow internal users to change some but not all of the attributes in their own entry. The directory administrators at `example.com` want to allow users to change their own password, home telephone number, and home address, but nothing else. This is illustrated in the ACI "Write example.com" example.

It is also `example.com`'s policy to let their subscribers update their own personal information in the `example.com` tree, provided that they establish an SSL connection to the directory. This is illustrated in the ACI "Write Subscribers" example.

ACI "Write example.com"

NOTE By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant `example.com` employees the right to update their password, home telephone number, and home address, you would write the following statement:

```
aci: (targetattr="userPassword || homePhone ||
homePostalAddress") (version 3.0; acl "Write example.com"; allow
(write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Write example.com`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area to Special Rights, and select Self from the Search results list.

   c. Click the Add button to list Self in the list of users who are granted access permission.

   d. Click OK to dismiss the Add Users and Groups dialog box.

4. In the Rights tab, select the checkbox for `write` right. Make sure the other checkboxes are clear.

5. In the Targets tab, click This Entry to display the `dc=example,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `userPassword` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.

6. In the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.example.com`. Click OK to dismiss the dialog box.

7. Click OK in the Access Control Editor window.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI "Write Subscribers"

NOTE By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant `example.com` subscribers the right to update their password and home telephone number, you would write the following statement:

```
aci: (targetattr="userPassword || homePhone") (version 3.0; acl
"Write Subscribers"; allow (write) userdn= "ldap://self" and
authmethod="ssl";)
```

This example assumes that the `aci` is added to the `ou=subscribers, dc=example,dc=com` entry.

`example.com` subscribers do not have `write` access to their home address because they might delete the attribute, and `example.com` needs that information for billing. Therefore, the home address is business-critical information.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the Subscribers entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Write Subscribers`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, tick the checkbox for `write`. Make sure the other checkboxes are clear.
5. In the Targets tab, click This Entry to display the `dc=subscribers, dc=example,dc=com` suffix in the target directory entry field.
   a. In the filter for subentries field, type the following filter:

      ```
      (!(unlistedSubscriber=yes))
      ```

   b. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

      All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.

6. If you want users to authenticate using SSL, switch to manual editing by clicking the Edit Manually button, and add `authmethod=ssl` to the LDIF statement so that it reads as follows:

   ```
   (targetattr="homePostalAddress || homePhone || mail")
   (version 3.0; acl "Write Subscribers"; allow (write) (userdn=
   "ldap:///self") and authmethod="ssl";)
   ```

7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Restricting Access to Key Roles

You can use role definitions in the directory to identify functions that are critical to your business, the administration of your network and directory, or another purpose.

For example, you might create a `superAdmin` role by identifying a subset of your system administrators that are available at a particular time of day and day of the week at corporate sites worldwide. Or you might want to create a `First Aid` role that includes all members of staff on a particular site that have done first aid training. For information on creating role definitions, refer to "Using Roles," on page 174.

When a role gives any sort of privileged user rights over critical corporate or business functions, you should consider restricting access to that role. For example, at `example.com`, employees can add any role to their own entry except the `superAdmin` role. This is illustrated in the ACI "Roles" example.

ACI "Roles"

In LDIF, to grant `example.com` employees the right to add any role to their own entry except the `superAdmin` role, you would write the following statement:

```
aci: (targetattr = "nsRoleDn")
(targattrfilters="add=nsRoleDN:(nsRoleDN !=
"cn=superAdmin,dc=example,dc=com")") (version 3.0; acl "Roles";
allow (write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Roles`. In the list of users granted access permission, do the following:

   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.

   c. Click the Add button to list Self in the list of users who are granted access permission.

   d. Click OK to dismiss the Add Users and Groups dialog box.

4. In the Rights tab, tick the checkbox for `write`. Make sure the other checkboxes are clear.

5. In the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.example.com`. Click OK to dismiss the dialog box.

6. To create the value-based filter for roles, switch to manual editing by clicking the Edit Manually button. Add the following to the beginning of the LDIF statement:

   ```
   (targattrfilters="add=nsRoleDN:(nsRoleDN != "cn=superAdmin,
   dc=example,dc=com")")
   ```

   The LDIF statement should read as follows:

   ```
   (targattrfilters="add=nsRoleDN:(nsRoleDN != "cn=superAdmin,
   dc=example,dc=com")") (targetattr = "*") (target =
   "ldap:///dc=example,dc=com") (version 3.0; acl "Roles"; allow
   (write) (userdn = "ldap:///self") and (dns="*.example.com");)
   ```

7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Granting a Group Full Access to a Suffix

Most directories have a group that is used to identify certain corporate functions. These groups can be given full access to all or part of the directory. By applying the access rights to the group, you can avoid setting the access rights for each member individually. Instead, you grant users these access rights simply by adding them to the group.

For example, when you install the Directory Server using the Typical Install process, an Administrators group with full access to the directory is created by default.

At `example.com`, the Human Resources group is allowed full access to the `ou=example-people` branch of the directory so that they can update the employee database. This is illustrated in the ACI "HR" example.

ACI "HR"

In LDIF, to grant the HR group all rights on the employee branch of the directory, you would use the following statement:

```
aci: (version 3.0; acl "HR"; allow (all) userdn=
"ldap:///cn=HRgroup,ou=example-people,dc=example,dc=com";)
```

This example assumes that the ACI is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `example.com-people` entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `HR`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area to Users and Groups, and type "HRgroup" in the Search for field.

      This example assumes that you have created an HR group or role. For more information on groups and roles, see chapter 5, "Advanced Entry Management."
   c. Click the Add button to list the HR group in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, click the Check All button.

   All checkboxes are ticked, except for Proxy rights.
5. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Granting Rights to Add and Delete Group Entries

Some organizations want to allow employees to create entries in the tree if it can increase their efficiency or if it can contribute to the corporate dynamics.

At `example.com` for example, there is an active social committee that is organized into various clubs: tennis, swimming, skiing, role-playing, etc. Any `example.com` employee can create a group entry representing a new club. This is illustrated in the ACI "Create Group" example. Any `example.com` employee can become a member of one of these groups. This is illustrated in ACI "Group Members" under "Allowing Users to Add or Remove Themselves from a Group," on page 264. Only the group owner can modify or delete a group entry. This is illustrated in the ACI "Delete Group" example.

ACI "Create Group"

In LDIF, to grant `example.com` employees the right to create a group entry under the `ou=Social Committee` branch, you would write the following statement:

```
aci: (target="ldap:///ou=social committee,dc=example,dc=com)
(targattrfilters="add=objectClass:(objectClass=groupOfNames)")
(version 3.0; acl "Create Group"; allow (add)
(userdn= "ldap:///uid=*,ou=example-people,dc=example,dc=com")
and dns="*.example.com";)
```

NOTE This ACI does not grant write permission, which means that the entry creator cannot modify the entry.

This example assumes that the ACI is added to the `ou=social committee, dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the Social Committee entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Create Group`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area to Special Rights, and select All Authenticated Users from the Search results list.
   c. Click the Add button to list All Authenticated Users in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.

4. In the Rights tab, tick the checkbox for `add`. Make sure the other checkboxes are clear.
5. In the Targets tab, click This Entry to display the `ou=social committee, dc=example,dc=com` suffix in the target directory entry field.
6. In the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.example.com`. Click OK to dismiss the dialog box.
7. To create the value-based filter that will allow employees to add only group entries to this subtree, switch to manual editing by clicking the Edit Manually button. Add the following to the beginning of the LDIF statement:

   ```
   (targattrfilters="add=objectClass:(objectClass=groupOfNames)"
    )
   ```

   The LDIF statement should read as follows:

   ```
   (targattrfilters="add=objectClass:(objectClass=groupOfNames)
   ") (targetattr = "*") (target="ldap:///ou=social
   committee,dc=example,dc=com) (version 3.0; acl "Create
   Group"; allow (read,search,add) (userdn= "ldap:///all") and
   (dns="*.example.com"); )
   ```

8. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI “Delete Group”

In LDIF, to grant `example.com` employees the right to modify or delete a group entry which they own under the `ou=Social Comittee` branch, you would write the following statement:

```
aci: (target="ou=social committee,dc=example,dc=com)
(targattrfilters="del=objectClass:(objectClass=groupOfNames)")
(version 3.0; acl "Delete Group"; allow (delete) userattr=
"owner#GROUPDN";)
```

This example assumes that the `aci` is added to the `ou=social committee, dc=example,dc=com` entry.

Using the Console is not an effective way of creating this ACI because you would have to use manual editing mode to create the target filter and to check group ownership.

Granting Conditional Access to a Group or Role

In many cases, when you grant a group or role privileged access to the directory, you want to ensure that those privileges are protected from intruders trying to impersonate your privileged users. Therefore, in many cases, access control rules that grant critical access to a group or role are often associated with a number of conditions.

`example.com`, for example, has created a Directory Administrator role for each of its hosted companies, `HostedCompany1` and `HostedCompany2`. It wants these companies to be able to manage their own data and implement their own access control rules while securing it against intruders. For this reason, `HostedCompany1` and `HostedCompany2` have full rights on their respective branches of the directory tree, provided the following conditions are fulfilled:

- Connection authenticated using SSL,
- Access requested between 8 a.m. and 6 p.m., Monday through Thursday, and
- Access requested from a specified IP address for each company.

These conditions are illustrated in a single ACI for each company, ACI “HostedCompany1” and ACI “HostedCompany2.” Because the content of these ACIs is the same, the examples below illustrate the “HostedCompany1” ACI only.

ACI “HostedCompany1”

In LDIF, to grant HostedCompany1 full access to their own branch of the directory under the conditions stated above, you would write the following statement:

```
aci:
(target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc=co
m")
(targetattr= "*") (version 3.0; acl "HostedCompany1";
allow (all)
(roledn="ldap:///cn=DirectoryAdmin,ou=HostedCompany1,
ou=corporate-clients, dc=example,dc=com") and
(authmethod="ssl") and
(dayofweek="Mon,Tues,Wed,Thu") and (timeofday >= "0800" and
timeofday <= "1800") and (ip="255.255.123.234"); )
```

This example assumes that the ACI is added to the `ou=HostedCompany1, ou=corporate-clients,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `HostedCompany1` entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `HostedCompany1`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area to Users and Groups, and type `DirectoryAdmin` in the Search For field.

      This example assumes that you have created an administrators role with a `cn` of `DirectoryAdmin`.
   c. Click the Add button to list the administrators role in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, click the Check All button.
5. In the Targets tab, click This Entry to display the `ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com` suffix in the target directory entry field.
6. In the Hosts tab, click Add to display the Add Host Filter dialog box. In the IP address host filter field, type `255.255.123.234`. Click OK to dismiss the dialog box.

   The IP address must be a valid IP address for the host machine that the `HostedCompany1` administrators will use to connect to the `example.com` directory.

7. In the Times tab, select the block time corresponding to Monday through Thursday and 8 a.m. to 6 p.m.

   A message appears below the table that specifies what time block you have selected.

8. To enforce SSL authentication from `HostedCompany1` administrators, switch to manual editing by clicking the Edit Manually button. Add the following to the end of the LDIF statement:

   ```
   and (authmethod="ssl")
   ```

   The LDIF statement should be similar to:

   ```
   aci: (targetattr = "*")
   (target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc
   =com") (version 3.0; acl "HostedCompany1"; allow (all)
   (roledn=
   "ldap:///cn=DirectoryAdmin,ou=HostedCompany1,ou=corporate-cli
   ents, dc=example,dc=com") and (dayofweek="Mon,Tues,Wed,Thu")
   and (timeofday >= "0800" and timeofday <= "1800") and
   (ip="255.255.123.234") and (authmethod="ssl"); )
   ```

9. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Denying Access

If your directory holds business-critical information, you might specifically want to deny access to it.

For example, `example.com` wants all subscribers to be able to read billing information such as connection time or account balance under their own entries but explicitly wants to deny write access to that information. This is illustrated in ACI “Billing Info Read” and ACI “Billing Info Deny,” respectively.

ACI “Billing Info Read”

In LDIF, to grant subscribers permission to read billing information in their own entry, you would write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
3.0; acl "Billing Info Read"; allow (search,read) userdn=
"ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the `ou=subscribers,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the subscribers entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Billing Info Read`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, tick the checkboxes for `search` and `read` rights. Make sure the other checkboxes are clear.
5. In the Targets tab, click This Entry to display the `ou=subscribers, dc=example,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `connectionTime` and `accountBalance` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.

   This example assumes that you have added the the `connectionTime` and `accountBalance` attributes to the schema.
6. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI “Billing Info Deny”

In LDIF, to deny subscribers permission to modify billing information in their own entry, you would write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
3.0; acl "Billing Info Deny"; deny (write) userdn=
"ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the `ou=subscribers,dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the subscribers entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Billing Info Deny`. In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, tick the checkbox for `write`. Make sure the other checkboxes are clear.
5. Click the Edit Manually button, and, in the LDIF statement that is displayed, change the word `allow` to `deny`.

6. In the Targets tab, click This Entry to display the `ou=subscribers, dc=example,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `connectionTime` and `accountBalance` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.

   This example assumes that you have added the the `connectionTime` and `accountBalance` attributes to the schema.

7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Setting a Target Using Filtering

If you want to set access controls that allow access to a number of entries that are spread across the directory, you may want to use a filter to set the target. Keep in mind that because search filters do not directly name the object for which you are managing access, it is easy to allow or deny access to the wrong objects unintentionally, especially as your directory becomes more complex. Additionally, filters can make it difficult for you to troubleshoot access control problems within your directory.

The following procedure shows you how to grant user `bjensen` write access to the department number, home phone number, home postal address, JPEG photo, and manager attributes for all members of the accounting organization.

Before you can set these permissions, you must create the accounting branch point (`ou=accounting,dc=example,dc=com`). You can create organizational unit branch points using the directory tab on the Directory Server Console.

Allowing Users to Add or Remove Themselves from a Group

Many directories set ACIs that allow users to add or remove themselves from groups. This is useful, for example, for allowing users to add and remove themselves from mailing lists.

At `example.com`, employees can add themselves to any group entry under the `ou=social committee` subtree. This is illustrated in the ACI "Group Members" example.

ACI "Group Members"

In LDIF, to grant `example.com` employees the right to add or delete themselves from a group, you would write the following statement:

```
aci: (targettattr="member")(version 3.0; acl "Group Members";
allow (selfwrite)
(userdn= "ldap:///uid=*,ou=example-people,dc=example,dc=com") ;)
```

This example assumes that the ACI is added to the `ou=social committee, dc=example,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. In the Directory tab, right click the `example-people` entry under the `example.com` node in the left navigation tree, and choose Set Access Permissions from the pop-up menu to display the Access Control Manager.
2. Click New to display the Access Control Editor.
3. In the Users/Groups tab, in the ACI name field, type `Group Members`. In the list of users granted access permission, do the following:
    a. Select and remove All Users, then click Add.

       The Add Users and Groups dialog box is displayed.
    b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select All Authenticated Users from the Search results list.
    c. Click the Add button to list All Authenticated Users in the list of users who are granted access permission.
    d. Click OK to dismiss the Add Users and Groups dialog box.
4. In the Rights tab, tick the checkbox for `selfwrite`. Make sure the other checkboxes are clear.
5. In the Targets tab, type `dc=example,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkbox for the `member` attribute.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.

6. Click OK.

The new ACI is added to the ones listed in the Access Control Manager window.

Defining Permissions for DNs That Contain a Comma

DNs that contain commas require special treatement within your LDIF ACI statements. In the target and bind rule portions of the ACI statement, commas must be escaped by a single backslash (\). The following example illustrates this syntax:

```
dn: dc=example.com Bolivia\, S.A.,dc=com
objectClass: top
objectClass: organization
aci: (target="ldap:///dc=example.com Bolivia\,
S.A.,dc=com")(targetattr=*) (version 3.0; acl "aci 2"; allow
(all)
groupdn = "ldap:///cn=Directory Administrators,dc=example.com
Bolivia\, S.A.,dc=com";)
```

Proxied Authorization ACI Example

For this example, suppose:

- The client application's bind DN is `"uid=MoneyWizAcctSoftware, ou=Applications,dc=example,dc=com"`.
- The targeted subtree to which the client application is requesting access is `ou=Accounting,dc=example,dc=com`.
- An Accounting Administrator with access permissions to the `ou=Accounting,dc=example,dc=com` subtree exists in the directory.

In order for the client application to gain access to the Accounting subtree (using the same access permissions as the Accounting Administrator):

- The Accounting Administrator must have access permissions to the `ou=Accounting,dc=example,dc=com` subtree. For example, the following ACI grants all rights to the Accounting Administrator entry:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com")
(targetattr="*") (version 3.0; acl "allowAll-AcctAdmin";
allow (all)
 userdn="ldap://uid=AcctAdministrator,ou=Administrators,dc=ex
ample,dc=com")
```

- The following ACI granting proxy rights to the client application must exist in the directory:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com")
(targetattr="*") (version 3.0; acl
"allowproxy-accountingsoftware"; allow (proxy)
userdn="ldap://uid=MoneyWizAcctSoftware,ou=Applications,dc=ex
ample,dc=com")
```

With this ACI in place, the `MoneyWizAcctSoftware` client application can bind to the directory and send an LDAP command such as `ldapsearch` or `ldapmodify` that requires the access rights of the proxy DN.

In the above example, if the client wanted to perform an `ldapsearch` command, the command would include the following controls:

```
#ldapmodify -D "uid=MoneyWizAcctSoftware,
ou=Applications,dc=example,dc=com" -w secretpwd
-y "uid=AcctAdministrator,ou=Administrators,dc=example,dc=com"
```

The client or application (`MoneyWizAcctSoftware`) binds as itself but is granted the privileges of the proxy entry (`AcctAdministrator`). The client does not need the password of the proxy entry.

NOTE You cannot use the directory manager's DN (Root DN) as a proxy DN. In addition, if Directory Server receives more than one proxied authentication control, an error is returned to the client application, and the bind attempt is unsuccessful.

Viewing the ACIs for an Entry

You can view all the ACIs under a single suffix in the directory by running the following `ldapsearch` command:

```
ldapsearch -h host -p port -b baseDN -D rootDN -w rootPassword (aci=*)
aci
```

See *Red Hat Directory Server Configuration, Command, and File Reference* for information on using the `ldapsearch` utility.

From the Console, you can view all of the ACIs that apply to a particular entry through the Access Control Manager.

1. In the Directory Console, on the Directory tab, right-click the entry in the navigation tree, and select Set Access Permissions.

   The Access Control Manager is displayed. It contains a list of the ACIs belonging to the selected entry.

2. Check the Show Inherited ACIs checkbox to display all ACIs created on entries above the selected entry that also apply.

Get Effective Rights Control

Finding the rights on existing attributes within a specific entry offers a convenient way for administrators to find and control the access rights.

"Get effective rights" is an extended `ldapsearch` which returns the access control permissions set on each attribute within an entry. The effective rights can be retrieved by sending an LDAP control along with a search operation. The results show the effective rights on each returned entry and each attribute of each returned entry.

The access control information is divided into two groups of access: rights for an entry and rights for an attribute. "Rights for an entry" means the rights, such as modify or delete, that are limited to that specific entry. "Rights for an attribute" means the access right to every instance of that attribute throughout the directory.

Some of the situations when this kind of detailed access control may be necessary include the following:

- An administrator can use the get effective rights command for minute access control, such as allowing certain groups or users access to entries and restricting others. For instance, members of the `QA Managers` group may have the right to search and read attributes like `manager` and `salary` but only `HR Group` members have the rights to modify or delete them.
- A user can run the get effective rights command to see what attributes he can view or modify on his personal entry. For instance, a user should have access to attributes such as `homePostalAddress` and `cn` but may only have read access to `manager` and `salary`.

An `ldapsearch` run with the `-J` tool will return the access controls placed on a particular entry. The `entryLevelRights` and `attributeLevelRights` returns are added as attributes to the bottom of the query results. If the `ldapsearch` is run without `-J`, then the entry information is returned as normal, without the `entryLevelRights` or `attributeLevelRights` information.

A get effective rights result looks like the following:

```
dn: uid=tmorris, ou=People, dc=example,dc=com
l: Santa Clara
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: vadn
attributeLevelRights: l:rscwo, userPassword:wo
```

In this example, Ted Morris has the right to add, view, delete, or rename the DN on his own entry, as shown by the returns in `entryLevelRights`. For attributes, he has the right to read, search, compare, self-modify, or self-delete the location (`l`) attribute but only self-write and self-delete rights to his password, as shown in the `attributeLevelRights` return.

Information is not given for attributes in an entry that do not have a value; for example, if the `userPassword` value is removed, then a future effective rights search on the entry above would not return any effective rights for `userPassword`, even though self-write and self-delete rights could be allowed. Likewise, if the `street` attribute were added with read, compare, and search rights, then `street: rsc` would appear in the `attributeLevelRights` results.

Table 6-3 and Table 6-4 summarize the permissions that can be set on entries and on attributes that are retrieved by the get effective rights operation.

Table 6-3 Permissions That Can Be Set on Entries

Permission	Description
`a`	Add.
`d`	Delete.
`n`	Rename the DN.
`v`	View the entry.

Table 6-4 Permissions That Can Be Set on Attributes

Permission	Description
`r`	Read.

Table 6-4 Permissions That Can Be Set on Attributes

Permission	**Description**
`s`	Search.
`w`	Write (`mod-add`).
`o`	Obliterate (`mod-del`). Analogous to delete.
`c`	Compare.
`W`	Self-write.
`O`	Self-delete.

Using Get Effective Rights from the Command-Line

To retrieve the effective rights with `ldapsearch`, you must pass the control information with the `ldapsearch` utility's `-J` option, as follows:

```
./ldapsearch -p port -h host -D bindDN -w bindPassword -b user -J control
OID:boolean criticality:dn:AuthId
```

- *user* specifies the account being checked, while *AuthId* checks the rights of the *AuthId* entry over the *user* entry.
- *control OID* is the OID for the get effective rights control, `1.3.6.1.4.1.42.2.27.9.5.2`.
- *boolean criticality* specifies whether the search operation should return an error if the server does not support this control (`true`) or if it should be ignored and let the search return as normal (`false`).
- *AuthId* is the DN of the entry whose rights over the *user* account are being checked. If the *AuthId* is left blank (`dn:`), than the rights of an anonymous user are returned.

A user, such as Ted Morris, can use this `ldapsearch` option to retrieve the rights he has to his personal entry, as shown below. Along with returning the effective rights information, the `ldapsearch` returns the regular entry information:

```
./ldapsearch -p 389 -h localhost -D
"uid=tmorris,ou=people,dc=example,dc=com" -w password -b
"uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:dn:
uid=tmorris,ou=people,dc=example,dc=com" "(objectClass=*)"
```

```
version: 1
dn: uid=tmorris, ou=People, dc=example,dc=com
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller, ou=People, dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: v
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc,
manager:rsc, roomNumber:rscwo, mail:rscwo,
facsimileTelephoneNumber:rscwo, objectClass:rsc, uid:rsc,
cn:rsc, userPassword:wo
```

An administrative user, such as Directory Manager, can use the get effective rights operation to determine what rights are granted between users. The following is a sample `ldapsearch` to retrieve effective rights that a manager, Dave Miller (shown in the `dn:`*user* part of the `-J` value), has over the entry of one of his subordinates, Ted Morris (shown in the `-b` value):

```
./ldapsearch -p 389 -h localhost -D "cn=directory manager" -w
password -b "uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:dn:
uid=dmiller,ou=people,dc=example,dc=com" "(objectClass=*)"
```

```
version: 1
dn: uid=tmorris, ou=People, dc=example,dc=com
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller, ou=People, dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
```

```
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: vadn
attributeLevelRights: givenName:rscwo, sn:rscwo, ou:rscwo,
l:rscwo, manager:rscwo, roomNumber:rscwo, mail:rscwo,
facsimileTelephoneNumber:rscwo, objectClass:rscwo, uid:rscwo,
cn:rscwo, userPassword:rscwo
```

For all attributes, Dave Miller has read, search, compare, modify, and delete permissions to Ted Morris's entry. These results are different than the ones returned in checking Ted Morris's access to his own entry, since he personally had only read, search, and compare rights to most of these attributes.

Only an administrator can retrieve effective rights to another user's entry. If Ted Morris tried to retrieve his rights to Dave Miller's entry, then he would receive the following error:

```
./ldapsearch -p 389 -h localhost -D
"uid=dmiller,ou=people,dc=example,dc=com" -w password -b
"uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:dn:
uid=tmorris,ou=people,dc=example,dc=com" "(objectClass=*)"

ldap_search: Insufficient access
ldap_search: additional info: get-effective-rights: requestor
has no g permission on the entry
```

However, Ted Morris could run a get effective rights search on his personal entry to determine the rights another user, such as Sam Carter, has to it. Assuming that an `ldapsearch` was run with `-b` set to `uid=tmorris,ou=people,dc=example,dc=com` and the *AuthId* was set to `uid=scarter,ou=people,dc=example,dc=com`, then Ted Morris would retrieve the following effective rights information:

```
entryLevelRights: v
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc,
manager:rsc, roomNumber:rsc, mail:rsc,
facsimileTelephoneNumber:rsc, objectClass:rsc, uid:rsc, cn:rsc,
userPassword:none
```

This means that Sam Carter has the right to view the DN of the entry and to read, search, and compare the `ou`, `givenName`, `l`, and other attributes and no rights to the `userPassword` attribute.

Using Get Effective Rights from the Console

To view effective rights from the Console, do the following:

1. Open the Directory tab, and right-click on the entry which rights you want to check.
2. Select “Advanced Properties” from the drop-down menu.

   The Properties Editor will appear.
3. Check the “Show effective rights” checkbox.

   The attribute-level effective rights (`r`, `s`, `c`, `w`, `o`) will appear next to the attributes. The entry-level rights (`v`, `a`, `d`, `n`) will appear under the full DN for the entry in the lower left-hand corner of the Properties Editor.

If you check the “Show all allowed attributes” checkbox, then the effective rights for those attributes will appear next to the additional attributes, even though they do not have values.

Get Effective Rights Return Codes

If the criticality is set to `false` for a get effective rights search and an error occurs, the regular entry information is returned, but, in place of rights for `entryLevelRights` and `attributeLevelRights`, an error code is returned. This code can give information on the configuration of the entry that was queried. Table 6-5 summarizes the error codes and the potential configuration information they can relay.

Table 6-5 Returned Result Codes

Code	Description
`0`	Successfully completed.
`1`	Operation error.
`12`	The critical extension is unavailable. If the criticality expression is set to `true` and effective rights do not exist on the entry being queried, then this error is returned.
`16`	No such attribute. If an attribute is specifically queried for access rights but that attribute does not exist in the schema, this error is returned.
`17`	Undefined attribute type.
`21`	Invalid attribute syntax.
`50`	Insufficient rights.

Table 6-5 Returned Result Codes

Code	Description
52	Unavailable.
53	Unwilling to perform.
80	Other.

Advanced Access Control: Using Macro ACIs

In organizations that use repeating directory tree structures, it is possible to optimize the number of ACIs used in the directory by using macros. Reducing the number of ACIs in your directory tree makes it easier to manage your access control policy and improves the efficiency of ACI memory usage.

Macros are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use a macro to represent a DN in the target portion of the ACI or in the bind rule portion, or both. In practice, when Directory Server gets an incoming LDAP operation, the ACI macros are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource. Directory Server then evaluates the ACI normally.

Macro ACI Example

The benefits of macro ACIs and how they work are best explained using an example. Figure 6-4, on page 275, shows a directory tree in which using macro ACIs is an effective way of reducing the overall number of ACIs.

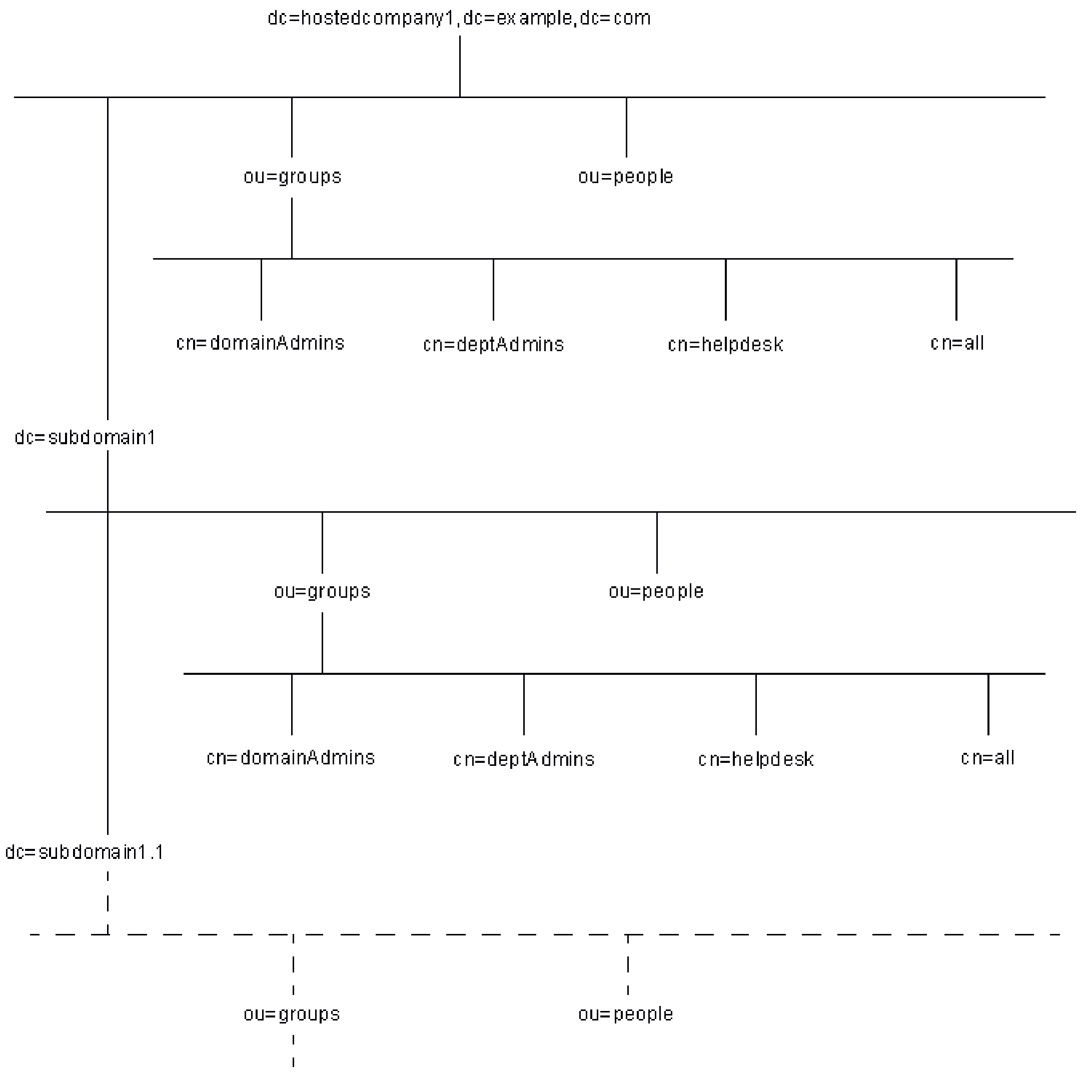
This illustration uses repeating pattern of subdomains with the same tree structure (`ou=groups`, `ou=people`). This pattern is also repeated across the tree because the `example.com` directory tree stores the suffixes `dc=hostedCompany2,dc=example,dc=com` and `dc=hostedCompany3,dc=example,dc=com`.

The ACIs that apply in the directory tree also have a repeating pattern. For example, the following ACI is located on the `dc=hostedCompany1,dc=example,dc=com` node:

```
aci:
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search) groupdn=
"ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example
,dc=com";)
```

This ACI grants `read` and `search` rights to the `DomainAdmins` group to any entry in the `dc=hostedCompany1,dc=example,dc=com` tree.

Figure 6-4 Example Directory Tree for Macro ACIs



The following ACI is located on the `dc=hostedCompany1,dc=example,dc=com` node:

```
aci:
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,
dc=example,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany1, dc=example,dc=com` node:

```
aci:
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,
dc=hostedCompany1,dc=example,dc=com";)
```

The following ACI is located on the `dc=hostedCompany2,dc=example,dc=com` node:

```
aci:
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany2,
dc=example,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany2, dc=example,dc=com` node:

```
aci:
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups, dc=subdomain1,
dc=hostedCompany2,dc=example,dc=com";)
```

In the four ACIs shown above, the only differentiator is the DN specified in the `groupdn` keyword. By using a macro for the DN, it is possible to replace these ACIs by a single ACI at the root of the tree, on the `dc=example,dc=com` node. This ACI reads as follows:

```
aci: (target="ldap:///ou=Groups,($dn),dc=example,dc=com")
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=
com";)
```

The target keyword, which was not previously used, needs to be introduced.

In the example above, the number of ACIs is reduced from four to one. However, the real benefit is a factor of how many repeating patterns you have down and across your directory tree.

Macro ACI Syntax

Macro ACIs include the following types of expressions to replace a DN or part of a DN:

- ($dn)
- [$dn]
- ($attr.*attrName*), where *attrName* represents an attribute contained in the target entry

To simplify the discussion in this section, the ACI keywords used to provide bind credentials, such as `userdn`, `roledn`, `groupdn`, and `userattr`, are collectively called the *subject*, as opposed to the *target*, of the ACI. Macro ACIs can be used in the target part or the subject part of an ACI.

Table 6-6 shows in what parts of the ACI you can use DN macros:

Table 6-6 Macros in ACI Keywords

Macro	ACI Keyword
`($dn)`	`target`, `targetfilter`, `userdn`, `roledn`, `groupdn`, `userattr`
`[$dn]`	`targetfilter`, `userdn`, `roledn`, `groupdn`, `userattr`
`($attr.`*attrName*`)`	`userdn`, `roledn`, `groupdn`, `userattr`

The following restrictions apply:

- If you use `($dn)` in `targetfilter`, `userdn`, `roledn`,`groupdn`, `userattr`, you *must* define a target that contains `($dn)`.
- If you use `[$dn]` in `targetfilter`, `userdn`, `roledn`,`groupdn`, `userattr`, you *must* define a target that contains `($dn)`.

In short, you when using any macro, you *always* need a target definition that contains the `($dn)` macro.

You can combine the `($dn)` macro and the `($attr.`*attrName*`)` macro.

Macro Matching for ($dn)

The `($dn)` macro is replaced by the matching part of the resource targeted in an LDAP request. For example, you have an LDAP request targeted at the `cn=all, ou=groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com` entry and an ACI that defines the target as follows:

```
(target="ldap:///ou=Groups,($dn),dc=example,dc=com")
```

The `($dn)` macro matches with `dc=subdomain1, dc=hostedCompany1`.

When the subject of the ACI also uses `($dn)`, the substring that matches the target is used to expand the subject. For example:

```
aci: (target="ldap:///ou=*,($dn),dc=example,dc=com")
(targetattr = "*") (version 3.0; acl "Domain access"; allow
(read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,($dn),dc=example,dc=
com";)
```

In this case, if the string matching `($dn)` in the target is `dc=subdomain1, dc=hostedCompany1`, then the same string is used in the subject. The ACI above is expanded as follows:

```
aci:
(target="ldap:///ou=Groups,dc=subdomain1,dc=hostedCompany1,
dc=example,dc=com") (targetattr = "*") (version 3.0; acl "Domain
access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,
dc=subdomain1,dc=hostedCompany1,dc=example,dc=com";)
```

Once the macro has been expanded, Directory Server evaluates the ACI following the normal process to determine whether access is granted.

Macro Matching for [$dn]

The matching mechanism for `[$dn]` is slightly different than for `($dn)`. The DN of the targeted resource is examined several times, each time dropping the left-most RDN component, until a match is found.

For example, you have an LDAP request targeted at the `cn=all,ou=groups, dc=subdomain1,dc=hostedCompany1,dc=example,dc=com` subtree and the following ACI:

```
aci: (target="ldap:///ou=Groups,($dn),dc=example,dc=com")
(targetattr = "*") (version 3.0; acl "Domain access"; allow
(read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=
com";)
```

The steps for expanding this ACI are as follows:

1. `($dn)` in target matches `dc=subdomain1,dc=hostedCompany1`.
2. Replace `[$dn]` in subject with `dc=subdomain1,dc=hostedCompany1`.

   The result is `groupdn="ldap:///cn=DomainAdmins,ou=Groups, dc=subdomain1,dc=hostedCompany1,dc=example,dc=com"`. If the bind DN is a member of that group, the matching process stops, and the ACI is evaluated. If it does not match, the process continues.
3. Replace `[$dn]` in subject with `dc=hostedCompany1`.

   The result is `groupdn="ldap:///cn=DomainAdmins,ou=Groups, dc=hostedCompany1,dc=example,dc=com"`. In this case, if the bind DN is not a member of that group, the ACI is not evaluated. If it is a member, the ACI is evaluated.

The advantage of the `[$dn]` macro is that it provides a flexible way of granting access to domain-level administrators to *all* the subdomains in the directory tree. Therefore, it is useful for expressing a hierarchical relationship between domains.

For example, consider the following ACI:

```
aci: (target="ldap:///ou=*, ($dn),dc=example,dc=com")
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=c
om";)
```

It grants access to the members of `cn=DomainAdmins,ou=Groups, dc=hostedCompany1,dc=example,dc=com` to all of the subdomains under `dc=hostedCompany1`, so an administrator belonging to that group could access; for example, the subtree `ou=people, dc=subdomain1.1, dc=subdomain1`.

However, at the same time, members of `cn=DomainAdmins,ou=Groups, dc=subdomain1.1` would be denied access to the `ou=people,dc=hostedCompany1` and `ou=people,dc=hostedCompany1` nodes.

Macro Matching for ($attr.*attrName*)

The `($attr.`*attrName*`)` macro is always used in the subject part of a DN. For example, you could define the following `roledn`:

```
roledn = "ldap:///cn=DomainAdmins,($attr.ou)"
```

Now, assume the server receives an LDAP operation targeted at the following entry:

```
dn: cn=Jane Doe, ou=People, dc=HostedCompany1, dc=example,dc=com
cn: Jane Doe
sn: Doe
ou: Engineering, dc=HostedCompany1, dc=example,dc=com
...
```

In order to evaluate the `roledn` part of the ACI, the server looks at the `ou` attribute stored in the targeted entry and uses the value of this attribute to expand the macro. Therefore, in the example, the `roledn` is expanded as follows:

```
roledn =
"ldap:///cn=DomainAdmins,ou=Engineering,dc=HostedCompany1,
dc=example,dc=com"
```

The Directory Server then evaluates the ACI according to the normal ACI evaluation algorithm.

When an attribute is multi-valued, each value is used to expand the macro, and the first one that provides a successful match is used.

Consider this example:

```
dn: cn=Jane Doe,ou=People,dc=HostedCompany1,dc=example,dc=com
cn: Jane Doe
sn: Doe
ou: Engineering, dc=HostedCompany1, dc=example,dc=com
ou: People, dc=HostedCompany1,dc=example,dc=com
...
```

In this case, when the Directory Server evaluates the ACI, it performs a logical OR on the following expanded expressions:

```
roledn =
"ldap:///cn=DomainAdmins,ou=Engineering,dc=HostedCompany1,
dc=example,dc=com"

roledn = "ldap:///cn=DomainAdmins,ou=People,dc=HostedCompany1,
dc=example,dc=com"
```

Access Control and Replication

ACIs are stored as attributes of entries; therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated like any other attribute.

ACIs are always evaluated on the Directory Server that services the incoming LDAP requests. This means that when a consumer server receives an update request, it will return a referral to the supplier server before evaluating whether the request can be serviced on the supplier.

Logging Access Control Information

To obtain information on access control in the error logs, you must set the appropriate log level.

To set the error log level from the Console:

1. In the Console, click the Directory tab, right click the config node, and choose Properties from the pop-up menu.

   This displays the Property Editor for the `cn=config` entry.

2. Scroll down the list of attribute value pairs to locate the `nsslapd-errorlog-level` attribute.

3. Add `128` to the value already displayed in the `nsslapd-errorlog-level` value field.

   For example, if the value already displayed is `8192` (replication debugging), you should change the value to `8320`. For complete information on error log levels, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

4. Click OK to dismiss the Property Editor.

Compatibility with Earlier Releases

Some ACI keywords that were used in earlier releases of Directory Server have been deprecated in release 6.1 and later. However, for reasons of backward compatibility, they are still supported. These keywords are:

- `userdnattr`
- `groupdnattr`

Therefore, if you have set up a replication agreement between a legacy supplier server and a consumer version later than 6.1, you should not encounter any problems in the replication of ACIs.

Chapter 7

User Account Management

When a user connects to your Red Hat Directory Server (Directory Server), first the user is authenticated. Then, the directory can grant access rights and resource limits to the user depending upon the identity established during authentication.

This chapter describes tasks for user account management, including configuring the password and account lockout policy for your directory, denying groups of users access to the directory, and limiting system resources available to users depending upon their bind DNs.

This chapter contains the following sections:

- Managing the Password Policy (page 283)
- Inactivating Users and Roles (page 300)
- Setting Resource Limits Based on the Bind DN (page 303)

NOTE For an overview on password policy, see section “Designing a Password Policy” in chapter 7, “Designing a Secure Directory,” in the *Red Hat Directory Server Deployment Guide*.

Managing the Password Policy

A password policy minimizes the risks of using passwords by enforcing the following:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords.

Once you have established a password policy, which can be for the entire directory or for specific subtrees or users, you can protect your user passwords from potential threats by configuring an account lockout policy. Account lockout protects against hackers who try to break into the directory by repeatedly guessing a user's password.

This section provides information about configuring your password and account lockout policies:

- Configuring the Password Policy
- Setting User Passwords
- Password Change Extended Operation
- Configuring the Account Lockout Policy
- Managing the Password Policy in a Replicated Environment
- Sycnhronizing Passwords

For an overview on password policy, check *Red Hat Directory Server Deployment Guide*.

Configuring the Password Policy

Directory Server supports fine-grained password policy, enabling you to define a policy that can be applied to the entire directory (*global* password policy), a particular subtree (*subtree level* or *local* password policy), or a particular user (*user level* or *local* password policy).

Essentially, your password policy is comprised of the following information:

- **The type or level of password policy checks.** This information indicates whether the server should check for and enforce a global password policy or local (subtree/user level) password policies.
- **Password add and modify information.** The password information includes password syntax and password history details.
- **Bind information.** The bind information includes the number of grace logins permitted, password aging attributes, and tracking bind failures.

The sections that follow describe the procedures for configuring your password policy:

- Configuring a Global Password Policy Using the Console

- Configuring a Subtree/User Password Policy Using the Console
- Configuring a Global Password Policy Using the Command-Line
- Configuring Subtree/User Password Policy Using the Command-Line

NOTE After configuring your password policy, we recommend that you configure an account lockout policy. For details, see "Configuring the Account Lockout Policy," on page 296.

Configuring a Global Password Policy Using the Console

To set up or modify the password policy for an entire directory:

1. In the Directory Server Console, select the Configuration tab and then the Data node.
2. In the right pane, select the Passwords tab.

   This tab contains the password policy for the entire Directory Server.
3. If you want users to change their password the first time they log on, select the "User must change password after reset" checkbox.

   If you select this checkbox, only the Directory Manager is authorized to reset the users's password. A regular administrative user cannot force the users to update their password.
4. If you want to allow users to change their own passwords, select the "User may change password" checkbox.
5. If you want to prevent users from changing their password for a specific duration, enter the number of days in the "Allow changes in X day(s)" text box.
6. If you want the server to maintain a history list of passwords used by each user, select the "Keep password history" checkbox. Enter the number of passwords you want the server to keep for each user in the "Remember X passwords" text box.
7. If you do not want user passwords to expire, select the "Password never expires" radio button.

8. If you want users to change their passwords periodically, select the “Password expires after X days” radio button, and then enter the number of days that a user password is valid.

   The maximum value for the password age is derived by subtracting January 18, 2038, from today’s date. The value you enter must not be set to the maximum value or too close to the maximum value. If you set the value to the maximum value, Directory Server may fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, you must correct the `passwordMaxAge` attribute value in the `dse.ldif` file.

   A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to `8640000` seconds (100 days).

9. If you have selected the “Password expire after X days” radio button, you need to specify how long before the password expires to send a warning to the user. In the “Send Warning X Days Before Password Expires” text enter the number of days before password expiration to send a warning.

10. If you want the server to check the syntax of a user password to make sure it meets the minimum requirements set by the password policy, select the “Check Password Syntax” checkbox. Then, specify the minimum acceptable password length in the “Password Minimum Length” text box.

11. From the “Password Encryption” pull-down menu, select the encryption method you want the server to use when storing passwords.

    For detailed information about the encryption methods, refer to the `passwordStorageScheme` attribute in Table 7-1, on page 287.

    The Password Encryption menu might contain other encryption methods, as the directory dynamically creates the menu depending upon the existing encryption methods it finds in your directory.

12. When you have finished making changes to the password policy, click Save.

Configuring a Subtree/User Password Policy Using the Console

To set up the password policy for a subtree or user, you need to add the required entries and attributes at the subtree or user level, set the appropriate values to the password policy attributes, and enable fine-grained password policy checking.

1. Enable fine-grained password policy.

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, select the Data node.

   c. In the right pane, select the Passwords tab.
   d. Check the "Enable fine-grained password policy" checkbox.
   e. Click Save to save your changes.

2. Create the local password policy for the subtree or user.
   a. In the Directory Server Console, select the Directory tab.
   b. In the navigation pane, select the subtree or user entry for which you want to set up the password policy.
   c. From the Object menu, select the Manage Password Policy option, and then select the "For user" or "For subtree."

      Depending on your selection, the User Password Policy or Subtree Password Policy window appears.
   d. In the Passwords tab, select the "Create subtree/user level password policy" checkbox to add the required attributes, fill in the appropriate values, and click Save.
   e. In the Account Lockout tab, specify the appropriate information, and click Save.

Configuring a Global Password Policy Using the Command-Line

This section describes the attributes you set to create a password policy for your entire server. Use `ldapmodify` to change these attributes in the `cn=config` entry.

Table 7-1 describes the attributes you can use to configure your password policy.

Table 7-1 Password Policy Attributes

Attribute Name	Definition
`passwordGraceLimit`	This attribute indicates the number of grace logins permitted when a user's password is expired. When set to a positive number, the user will be allowed to bind with the expired password for that many times. For the global password policy, the attribute is defined under `cn=config`. By default, this attribute is set to `0`, which means grace logins are not permitted.

Table 7-1 Password Policy Attributes *(Continued)*

Attribute Name	Definition
`passwordMustChange`	When `on`, this attribute requires users to change their passwords when they first login to the directory or after the password is reset by the Directory Manager. When `on`, the user is required to change their password even if user-defined passwords are disabled. If you choose to set this attribute to `off`, passwords assigned by the Directory Manager should not follow any obvious convention and should be difficult to discover. This attribute is `off` by default.
`passwordChange`	When `on`, this attribute indicates that users may change their own password. Allowing for users to set their own passwords runs the risk of users choosing passwords that are easy to remember. However, setting good passwords for the user requires a significant administrative effort. In addition, providing passwords to users that are not meaningful to them runs the risk that users will write the password down somewhere that can be discovered. This attribute is `on` by default.
`passwordExp`	When `on`, this attribute indicates that the user's password will expire after an interval given by the `passwordMaxAge` attribute. Making passwords expire helps protect your directory data because the longer a password is in use, the more likely it is to be discovered. This attribute is `off` by default.
`passwordMaxAge`	This attribute indicates the number of seconds after which user passwords expire. To use this attribute, you must enable password expiration using the `passwordExp` attribute. This attribute is a dynamic parameter in that its maximum value is derived by subtracting January 18, 2038, from today's date. The attribute value must not be set to the maximum value or too close to the maximum value. If you set the value to the maximum value, Directory Server may fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, you must correct the `passwordMaxAge` attribute value in the `dse.ldif` file. A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to `8640000` seconds (100 days).

Table 7-1 Password Policy Attributes *(Continued)*

Attribute Name	Definition
passwordWarning	This attribute indicates the number of seconds before a warning message is sent to users whose password is about to expire. Depending on the LDAP client application, users may be prompted to change their password when the warning is sent. Both Red Hat Directory Express and the Directory Server Gateway provide this functionality. By default, the directory sends the warning 86400 seconds (1 day) before the password is about to expire. However, a password never expires until the warning message has been set. Therefore, if users don't bind to the Directory Server for longer than the passwordMaxAge, they will still get the warning message in time to change their password.
passwordCheckSyntax	When on, this attribute indicates that the password syntax will be checked by the server before the password is saved. Password syntax checking ensures that the password string meets or exceeds the minimum password length requirements and that the string does not contain any "trivial" words. A trivial word is any value stored in the uid, cn, sn, givenName, ou, or mail attributes of the user's entry. This attribute is off by default.
passwordMinLength	This attribute specifies the minimum number of characters that must be used in passwords. Shorter passwords are easier to crack. You can require passwords that are 2 to 512 characters long. Generally, a length of 6 to 8 characters is long enough to be difficult to crack but short enough for users to remember without writing it down. This attribute is set to 6 by default.
passwordMinAge	This attribute indicates the number of seconds that must pass before a user can change their password. Use this attribute in conjunction with the passwordInHistory attribute to discourage users from reusing old passwords. For example, setting the minimum password age to 2 days prevents users from repeatedly changing their passwords during a single session to cycle through the password history and reuse an old password once it has been removed from the history list. You can specify from 0 to 2147472000 seconds (24,855 days). A value of zero indicates that the user can change the password immediately. The default value of this attribute is 0.

Table 7-1 Password Policy Attributes *(Continued)*

Attribute Name	**Definition**
`passwordHistory`	This attribute indicates whether the directory stores a password history. When set to `on`, the directory stores the number of passwords you specify in the `passwordInHistory` attribute in a history. If a user attempts to reuse one of the passwords, the password will be rejected. When you set this attribute to `off`, any passwords stored in the history remain there. When you set this attribute back to `on`, users will not be able to reuse the passwords recorded in the history before you disabled the attribute. This attribute is `off` by default, meaning users can reuse old passwords.
`passwordInHistory`	This attribute indicates the number of passwords the directory stores in the history. You can store from `2` to `24` passwords in the history. This feature is not enabled unless the `passwordHistory` attribute is set to `on`. This attribute is set to `6` by default.
`passwordStorageScheme`	This attribute specifies the type of encryption used to store Directory Server passwords. The following encryption types are supported by Directory Server: • `SSHA` (Salted Secure Hash Algorithm). This method is recommended as it is the most secure. This is the default method. • `SHA` (Secure Hash Algorithm). A one-way hash algorithm; it is supported only forbackwards compatibility with Directory Server 4.x and should not be used otherwise. • `crypt`. The UNIX crypt algorithm, provided for compatibility with UNIX passwords. • `clear`. This encryption type indicates that the password will appear in plain text. Passwords stored using `crypt`, `SHA`, or `SSHA` formats cannot be used for secure login through SASL Digest MD5. If you want to provide your own customized storage scheme, consult Red Hat Professional Services.

Configuring Subtree/User Password Policy Using the Command-Line

To configure a subtree or user level password policy:

1. Add the required attributes to the subtree or user entries by running the `ns-newpwpolicy.pl` script.

The command syntax for the script is as follows:

```
ns-newpwpolicy.pl [-D rootDN] { -w password | -w - | -j filename }
[-p port] [-h host] -U userDN -S suffixDN
```

For updating a subtree entry, use the `-S` option. For updating a user entry, use the `-U` option. The `ns-newpwpolicy.pl` script accepts only one user or subtree entry at a time. It can, however, accept both user and suffix entries at the same time. For details about the script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

The script adds the required attributes depending on whether the target entry is a subtree or user entry.

For a subtree (for example, `ou=people, dc=example, dc=com`), the following entries are added:

- A container entry (`nsPwPolicyContainer`) at the subtree level for holding various password policy-related entries for the subtree and all its children. For example:

```
dn: cn=nsPwPolicyContainer, ou=people, dc=example, dc=com
objectClass: top
objectClass: nsContainer
cn: nsPwPolicyContainer
```

- The actual password policy specification entry (`nsPwPolicyEntry`) for holding all the password policy attributes that are specific to the subtree. For example:

```
dn: cn="cn=nsPwPolicyEntry, ou=people, dc=example,
dc=com", cn=nsPwPolicyContainer, ou=people, dc=example,
dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

- The CoS template entry (`nsPwTemplateEntry`) that has the `pwdpolicysubentry` value pointing to the above (`nsPwPolicyEntry`) entry. For example:

```
dn: cn="cn=nsPwTemplateEntry, ou=people, dc=example,
dc=com", cn=nsPwPolicyContainer, ou=people, dc=example,
dc=com
objectclass: top
objectclass: extensibleObject
objectclass: costemplate
objectclass: ldapsubentry
cosPriority: 1
pwdpolicysubentry: cn="cn=nsPwPolicyEntry, ou=people,
dc=example, dc=com", cn=nsPwPolicyContainer, ou=people,
dc=example, dc=com
```

- The CoS specification entry at the subtree level. For example:

```
dn: cn=nsPwPolicy_cos, ou=people, dc=example, dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn="cn=nsPwTemplateEntry, ou=people,
dc=example, dc=com", cn=nsPwPolicyContainer, ou=people,
dc=example, dc=com
cosAttribute: pwdpolicysubentry default operational
```

For a user (for example, `uid=jdoe, ou=people, dc=example, dc=com`), the following entries are added:

- A container entry (`nsPwPolicyContainer`) at the parent level for holding various password policy related entries for the user and all its children. For example:

```
dn: cn=nsPwPolicyContainer, ou=people, dc=example, dc=com
objectClass: top
objectClass: nsContainer
cn: nsPwPolicyContainer
```

- The actual password policy specification entry (`nsPwPolicyEntry`) for holding the password policy attributes that are specific to the user. For example:

```
dn: cn="cn=nsPwPolicyEntry, uid=jdoe, ou=people,
dc=example, dc=com", cn=nsPwPolicyContainer, ou=people,
dc=example, dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

- Assign the value of the above entry DN to the `pwdpolicysubentry` attribute of the target entry. For example:

```
dn: uid=jdoe, ou=people, dc=example, dc=com
changetype: modify
replace: pwdpolicysubentry
pwdpolicysubentry: "cn=nsPwPolicyEntry, uid=jdoe,
ou=people, dc=example, dc=com", cn=nsPwPolicyContainer,
ou=people, dc=example, dc=com
```

2. Set the password policy attributes of subtree or user entry with the appropriate values.

   Table 7-1 describes the attributes you can use to configure your password policy. You may use the `ldapmodify` utility to change these attributes in the `cn=config` entry.

NOTE The `nsslapd-pwpolicy-local` attribute of the `cn=config` entry controls the type of password policy the server enforces. By default, this attribute is disabled (`off`). When the attribute is disabled, the server only checks for and enforces the global password policy; the subtree and user level password policies are ignored.

When you run the `ns-newpwpolicy.pl` script, it first checks for the specified subtree and user entries and, if they exist, modifies them. After updating the entries successfully, the script sets the `nsslapd-pwpolicy-local` configuration parameter to `on`.

If you don't want to enable the subtree and user level password policy, be sure to set `nsslapd-pwpolicy-local` to `off` after you run the script.

To turn off user and subtree level password policy checks, set the `nsslapd-pwpolicy-local` attribute to `off` by modifying the `cn=config` entry. For example, you can use the `ldapmodify` command to make these changes:

```
dn: cn=config
changetype: modify
replace: nsslapd-pwpolicy-local: on
nsslapd-pwpolicy-local: off
```

You can also disable the attribute by modifying it directly in the configuration file (`dse.ldif`). To do this:

1. Stop the server.
2. Open the `dse.ldif` file in a text editor.
3. Set the value of `nsslapd-pwpolicy-local` to `off`, and save your changes.
4. Start the server.

Setting User Passwords

An entry can be used to bind to the directory only if it has a `userpassword` attribute and if it has not been inactivated. Because user passwords are stored in the directory, you can use whatever LDAP operation you normally use to update the directory to set or reset the user passwords.

For information on creating and modifying directory entries, see chapter 2, "Creating Directory Entries." For information on inactivating user accounts, refer to "Inactivating Users and Roles," on page 300.

You can also use the Users and Groups area of the Red Hat Administration Server or the Directory Server Gateway to set or reset user passwords. For information on how to use the Users and Groups area, see the online help that is available in the Red Hat Administration Server. For information on how to use the Gateway to create or modify directory entries, see the online help that is available in the Gateway.

Password Change Extended Operation

While most passwords can be changed through the Console and other Directory Server features or through the `ldapmodify` operation, there are some passwords that cannot be changed through regular LDAP operations. These passwords may be stored outside the Directory Server, such as passwords stored in a SASL application. These passwords can be modified through the password change extended operation.

Directory Server supports the password change extended operation as defined in RFC 3062. This allows users to change their passwords, using a suitable client, in a standards-compliant way. Directory Server does not include a client application for the password change extended operation. However, the `ldappasswd` utility from OpenLDAP can be used as follows:

```
./ldappasswd -H ldaps://hostname:port -ZZ[Z] -D bindDN -w bindPassword
-a oldPassword -s newPassword user
```

For more information on how to use `ldappasswd` utility, see the OpenLDAP documentation at `http://www.openldap.org`, or type `man ldappasswd` in the command-line for the `ldappasswd` manpage.

NOTE This operation supports Start TLS encryption (`-ZZ[Z]`), and you must use a secure connection for the password change operation.

NOTE If your certificates are either self-signed or are issued by a certificate authority not trusted by the client application, then you may need to create a configuration file which contains the option `TLS_REQCERT never`, which suppresses certificate verification, or `TLS_CACERT /path/to/cacert.pem`, which specifes the path to you CA certificate. Set the `LDAPConf` environment variable to this file.

To modify an entry's password, run `ldappasswd` like any other LDAP operation. It is not necessary to specify a *user* if the account is the same as that given in the bindDN. For example:

```
./ldappasswd -H ldaps://server.example.com:24256 -ZZ -D
"uid=jsmith,ou=People,dc=example,dc=com" -w oldpassword -a
oldpassword -s newpassword
```

To change the password on an entry other than the one specified in the bind credentials, run `ldappasswd` as shown below, adding the *user* DN to the operation and providing separate credentials, as follows:

```
ldappasswd -H ldaps://server.example.com:24256 -ZZ -D
"cn=Directory Manager" -w rootpassword -a oldpassword -s
newpassword "uid=jsmith,ou=People,dc=example,dc=com"
```

Access control is enforced for the password change operation. If the bindDN does not have rights to change the specified password, the operation will fail with the "Insufficient rights" error.

Configuring the Account Lockout Policy

The lockout policy works in conjunction with the password policy to provide further security. The account lockout feature protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. You can set up your password policy so that a specific user is locked out of the directory after a given number of failed attempts to bind.

Configuring the account lockout policy is described in the following sections:

- Configuring the Account Lockout Policy Using the Console
- Configuring the Account Lockout Policy Using the Command-Line

Configuring the Account Lockout Policy Using the Console

To set up or modify the account lockout policy for your Directory Server:

1. In the Directory Server Console, select the Configuration tab and then the Data node.
2. In the right pane, select the Account Lockout tab.
3. To enable account lockout, select the "Accounts may be locked out" checkbox.
4. Enter the maximum number of allowed bind failures in the "Lockout account after X login failures" text box. The server locks out users who exceed the limit you specify here.
5. Enter the number of minutes you want the server to wait before resetting the bind failure counter to 0 in the "Reset failure counter after X minutes" text box.

6. Set the interval you want users to be locked out of the directory.

   Select the Lockout Forever radio button to lock users out until their passwords have been reset by the administrator.

   Set a specific lockout period by selecting the Lockout Duration radio button and entering the time (in minutes) in the text box.

7. When you have finished making changes to the account lockout policy, click Save.

Configuring the Account Lockout Policy Using the Command-Line

This section describes the attributes you set to create an account lockout policy to protect the passwords stored in your server. Use `ldapmodify` to change these attributes in the `cn=config` entry.

Table 7-2 describes the attributes you can use to configure your account lockout policy.

Table 7-2 Account Lockout Policy Attributes

Attribute Name	Definition
`passwordLockout`	This attribute indicates whether users are locked out of the directory after a given number of failed bind attempts. You set the number of failed bind attempts after which the user will be locked out using the `passwordMaxFailure` attribute. You can lock users out for a specific time or until an administrator resets the password. This attribute is set to `off` by default, meaning that users will not be locked out of the directory.
`passwordMaxFailure`	This attribute indicates the number of failed bind attempts after which a user will be locked out of the directory. This attribute takes affect only if the `passwordLockout` attribute is set to `on`. This attribute is set to `3` bind failures by default.
`passwordLockoutDuration`	This attribute indicates the time, in seconds, that users will be locked out of the directory. You can also specify that a user is locked out until the password is reset by an administrator using the `passwordUnlock` attribute. By default, the user is locked out for `3600` seconds.

Table 7-2 Account Lockout Policy Attributes *(Continued)*

Attribute Name	Definition
`passwordResetFailureCount`	This attribute specifies the time, in seconds, after which the password failure counter will be reset. Each time an invalid password is sent from the user's account, the password failure counter is incremented. If the `passwordLockout` attribute is set to `on`, users will be locked out of the directory when the counter reaches the number of failures specified by the `passwordMaxFailure` attribute. The account is locked out for the interval specified in the `passwordLockoutDuration` attribute, after which time the failure counter is reset to zero (`0`). Because the counter's purpose is to gauge when a hacker is trying to gain access to the system, the counter must continue for a period long enough to detect a hacker. However, if the counter were to increment indefinitely over days and weeks, valid users might be locked out inadvertently. The reset password failure count attribute is set `600` seconds by default.

Managing the Password Policy in a Replicated Environment

Password and account lockout policies are enforced in a replicated environment as follows:

- Password policies are enforced on the data master.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy information in your directory is replicated. The replicated attributes are:

- `passwordMinAge` and `passwordMaxAge`
- `passwordExp`
- `passwordWarning`

However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications. Account lockout counters and tiers are not replicated, either.

When configuring a password policy in a replicated environment, consider the following points:

- Warnings from the server of an impending password expiration will be issued by all replicas. This information is kept locally on each server, so if a user binds to several replicas in turn, they will be issued the same warning several times. In addition, if the user changes the password, it may take time for this information to filter to the replicas. If a user changes a password and then immediately rebinds, he may find that the bind fails until the replica registers the changes.
- You want the same bind behavior to occur on all servers, including suppliers and replicas. Make sure to create the same password policy configuration information on each server.
- Account lockout counters many not work as expected in a multi-mastered environment.
- Entries that are created for replication (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the `passwordExpirationTime` attribute to the entry, and give it a value of `20380119031407Z` (the top of the valid range).

Sycnhronizing Passwords

Password changes in a Directory Server entry can be synchronized to password attributes in Windows NT4 server or Active Directory entries by using the Password Sync utility.

When passwords are synchronized, password policies are enforced on each sync peer locally; i.e., the syntax or minimum length requirements on the Directory Server apply when the password is changed in the Directory Server; when the changed password is synched over to the Windows server, the Winodws password policy is enforced. The password policies themselves are not synchronized.

Configuration information is kept locally and cannot be synchronized. This information includes the history of password modifications. Account lockout counters are not synchronized, either.

When configuring a password policy for synchornization, consider the following points:

- The Password Sync utility must be installed locally on the Windows machine that will be synchronized with a Directory Server.

- Password Sync can only link the Windows machine to a single Directory Server; to sync changes with multiple Directory Server, configure the Directory Server for multi-master replication.
- Password expiration warnings and times, failed bind attempts, and other password-related information is enforced locally per server adn is not synchronized between sync peer servers.
- You want the same bind behavior to occur on all servers. Make sure to create the same or similar password policies on both Directory Server, Windows NT4 Server, and Active Directory servers.
- Entries that are created for synchronization (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the `passwordExpirationTime` attribute to the Directory Server entry, and give it a value of `20380119031407Z` (the top of the valid range).

See chapter 18, "Windows Sync," for more information on synchronizing Directory Server and Windows users and passwords.

Inactivating Users and Roles

You can temporarily inactivate a single user account or a set of accounts. Once inactivated, a user cannot bind to the directory. The authentication operation will fail.

Users and roles are inactivated using the operational attribute `nsAccountLock`. When an entry contains the `nsAccountLock` attribute with a value of `true`, the server rejects the bind.

You use the same procedures for inactivating users and roles. However, when you inactivate a role, you are inactivating the members of the role and not the role entry itself. For more information about roles in general and how roles interact with access control in particular, refer to chapter 5, "Advanced Entry Management."

The rest of this section describes the following procedures:

- Inactivating User and Roles Using the Console
- Inactivating User and Roles Using the Command-Line
- Activating User and Roles Using the Console
- Activating User and Roles Using the Command-Line

CAUTION You cannot inactivate the root entry (the entry corresponding to the root or sub suffix) on a database. For more information on creating the entry for a root or sub suffix, refer to chapter 2, "Creating Directory Entries," for more information. For more information on creating root and sub suffixes, refer to chapter 3, "Configuring Directory Databases."

Inactivating User and Roles Using the Console

The following procedure describes inactivating a user or a role using the Console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role you want to inactivate.

   The Edit Entry dialog box appears.

   You can also select Inactivate from the Object menu as a short cut.
3. Click Account in the left pane. The right pane states that the role or user is inactivated. Click Activate to activate the user or role.
4. Click OK to close the dialog box and save your changes.

   Once inactivated, you can view the state of the object by selecting Inactivation State from the View menu. The icon of the object then appears in the right pane of the Console with a red slash through it.

Inactivating User and Roles Using the Command-Line

To inactivate a user account, use the `ns-inactivate.pl` script. The following example describes using the `ns-inactivate.pl` script to inactivate Joe Frasier's user account:

```
ns-inactivate.pl -D "Directory Manager" -w secretpwd -p 389
-h example.com -I "uid=jfrasier,ou=people,dc=example,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
`-D`	The DN of the directory administrator.
`-w`	The password of the directory administrator.
`-p`	Port used by the server.
`-h`	Name of the server on which the directory resides.
`-I`	DN of the user account or role you want to inactivate.

For more information about running the `ns-inactivate.pl` script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

Activating User and Roles Using the Console

The following procedure describes activating a user or a role using the Console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role you want to activate.

   The Edit Entry dialog box appears.

   You can also select Activate from the Object menu as a short cut.
3. Click Account in the left pane. The right pane states that the role or user is activated. Click Activate to activate the user or role.
4. If the user or role is a member of another inactivated role, the Console displays an option for viewing the inactivated roles. Click Show Inactivated Roles to view the list of roles to which the user or role belongs.
5. Click OK when you are finished.

   Once reactivated, you can view the state of the object by selecting Inactivation State from the View menu. The icon of the role or user in the right pane of the Console appears as normal. The red slash through the icon indicating it was inactive disappears.

Activating User and Roles Using the Command-Line

To activate a user account, use the `ns-activate.pl` script. The following example describes using the `ns-activate.pl` script to activate Joe Frasier's user account:

```
ns-activate.pl -D "Directory Manager" -w secretpwd -p 389
-h example.com -I "uid=jfrasier,ou=people,dc=example,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
`-D`	The DN of the directory administrator.
`-w`	The password of the directory administrator.
`-p`	Port used by the server.
`-h`	Name of the server on which the directory resides.
`-I`	DN of the user account or role you want to activate.

For more information about running the `ns-activate.pl` script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

Setting Resource Limits Based on the Bind DN

You can control server limits for search operations 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.

NOTE The Directory Manager receives unlimited resources by default.

The resource limits you set for the client application take precedence over the default resource limits you set for in the global server configuration.

This section gives procedures for the following:

- Setting Resource Limits Using the Console
- Setting Resource Limits Using the Command-Line

Setting Resource Limits Using the Console

The following procedure describes setting resource limits for a user or a role using the Console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role for which you want to set resource limits.

   The Edit Entry dialog box appears.
3. Click Account in the left pane. The right pane contains the four limits you can set in the Resource Limits section.

   Entering a value of `-1` indicates no limit.
4. Click OK when you are finished.

Setting Resource Limits Using the Command-Line

The following operational attributes can be set for each entry using the command-line. Use `ldapmodify` to add the following attributes to the entry:

Attribute	Description
`nsLookThroughLimit`	Specifies how many entries examined for a search operation. Specified as a number of entries. Giving this attribute a value of `-1` indicates that there is no limit.

Attribute	Description
`nsSizeLimit`	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.
`nsTimeLimit`	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`	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.

For example, you might set the size limit for an entry by performing an `ldapmodify` as follows:

```
ldapmodify -h myserver -p 389 -D "cn=directory manager" -w
secretpwd
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.

Chapter 8

Managing Replication

Replication is the mechanism by which directory data is automatically copied from one Red Hat Directory Server (Directory Server) to another; it is an important mechanism for extending your directory service beyond a single server configuration. This chapter describes the tasks to be performed on the supplier servers and the consumer servers to set up single-master replication, multi-master replication, and cascading replication. This chapter includes the following topics:

For conceptual information on how you can use replication in your directory deployment, see the *Red Hat Directory Server Deployment Guide*.

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. This section contains information on the following replication concepts:

- Read-Write Replica/Read-Only Replica
- Supplier/Consumer
- Changelog
- Unit of Replication
- Replication Identity
- Replication Agreement
- Compatibility with Earlier Versions of Directory Server

Read-Write Replica/Read-Only Replica

A database that participates in replication is defined as 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 refers all update operations to read-write replicas. A server can hold any number of read-only or read-write replicas.

Supplier/Consumer

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. There are exceptions to this statement:

- In the case of cascading replication, the hub supplier holds a read-only replica that it supplies to consumers. For more information, refer to "Cascading Replication," on page 316.
- In the case of multi-master replication, the masters are suppliers and consumers for the same read-write replica. For more information, refer to "Multi-Master Replication," on page 313.

In Directory Server, replication is always initiated by the supplier server, never by the consumer. This operation is called supplier-initiated replication. It allows you to configure a supplier server to push data to one or more consumer servers.

Earlier versions of the Directory Server allowed consumer-initiated replication, where you could configure consumer servers to pull data from a supplier server.

Changelog

Every supplier server maintains a changelog. A changelog is a record 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.

In Directory Server, the format of the changelog has changed. It is now only intended for internal use by the server. If you have applications that need to read the changelog, you need to use the Retro Changelog Plug-in for backward compatibility. For more information, refer to "Using the Retro Changelog Plug-in," on page 363.

Unit of Replication

The smallest unit of replication is a database. This means that you can replicate an entire database but not a subtree within a database. Therefore, when you create your directory tree, you must take your replication plans into consideration. For more information on how to set up your directory tree, refer to the *Red Hat Directory Server Deployment Guide*.

The replication mechanism also requires that one database correspond to one suffix. This means that you cannot replicate a suffix (or namespace) that is distributed over two or more databases using custom distribution logic. For more information on this topic, refer to “Creating and Maintaining Databases,” on page 92.

Replication Identity

When replication occurs between two servers, the replication process uses a special entry, often referred to as the Replication Manager entry, to identify replication protocol exchanges. The Replication Manager entry, or any entry you create to fulfill that role, must meet the following criteria:

- It is created on the consumer server (or hub supplier) and *not* on the supplier server.
- You must create this entry on *every* server that receives updates from another server, meaning on every hub supplier or dedicated consumer.
- When you configure a replica that receives updates from another server, you must specify this entry as the one authorized to perform replication updates.
- When you configure the replication agreement on the supplier server, you must specify the DN of this entry in the replication agreement.
- This 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.

NOTE In the Directory Server Console, this Replication Manager entry is referred to as the *supplier bind DN*, which may be misleading as the entry does not actually exist on the supplier server. It is called the supplier bind DN because it is the entry which must be present on the consumer for the supplier to be able to bind to the consumer.

For more information on creating the Replication Manager entry, refer to "Creating the Supplier Bind DN Entry," on page 318.

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. It specifies:

- The database to be replicated.
- The consumer server to which the data is pushed.
- The times during which replication can occur.
- The DN and credentials that the supplier server must use to bind (called the Replication Manager entry or supplier bind DN).
- How the connection is secured (SSL, client authentication).
- Any attributes that will not be replicated. (For information on fractional replication, refer to the *Red Hat Directory Server Deployment Guide*.)

Compatibility with Earlier Versions of Directory Server

The replication mechanism in current versions of Directory Server is different from the mechanism used in earlier versions (4.x) of Directory Server. Compatibility is provided through the following:

- Legacy Replication Plug-in — The Legacy Replication Plug-in makes Directory Server behave as a 4.x Directory Server in a consumer role. For information on how to implement legacy replication using this plug-in, refer to "Replication with Earlier Releases," on page 361.
- Retro Changelog Plug-in — The Retro Changelog Plug-in can be used when you want 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, refer to "Using the Retro Changelog Plug-in," on page 363.

Replication Scenarios

This section describes the most commonly used replication scenarios:

- Single-Master Replication
- Multi-Master Replication
- Cascading Replication

You can combine these basic scenarios to build the replication environment that best suits your needs.

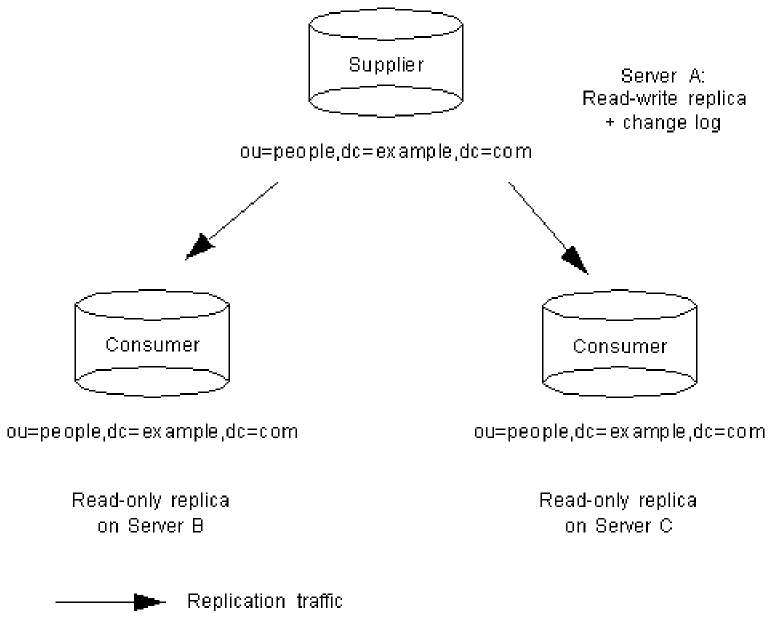
NOTE Whatever replication scenario you choose to implement, remember to consider schema replication. For details, see *Red Hat Directory Server Deployment Guide*.

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.

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, you have as many read-only replicas as you like. Such scenarios are called single-master configurations. Figure 8-1 shows an example of single-master replication.

Figure 8-1 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, refer to "Configuring Single-Master Replication," on page 326.

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.

This type of configuration can work with any number of consumer servers. Each consumer server holds a read-only replica. The consumers can receive updates from 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 8-2 shows an example of multi-master replication scenario with two supplier servers and two consumer servers.

Figure 8-2 Multi-Master Replication (Two Suppliers)

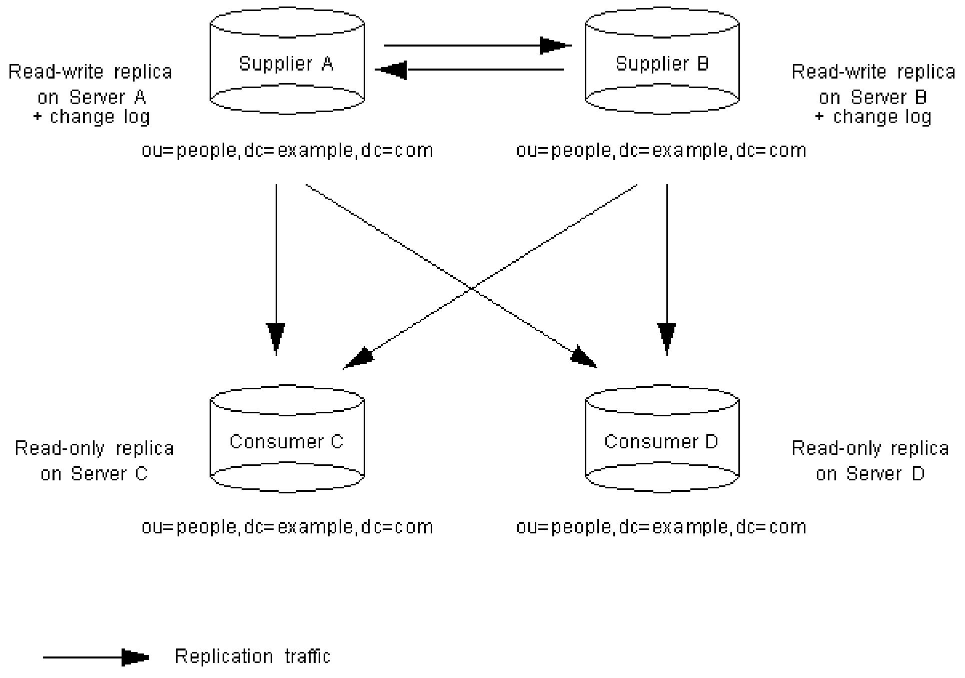
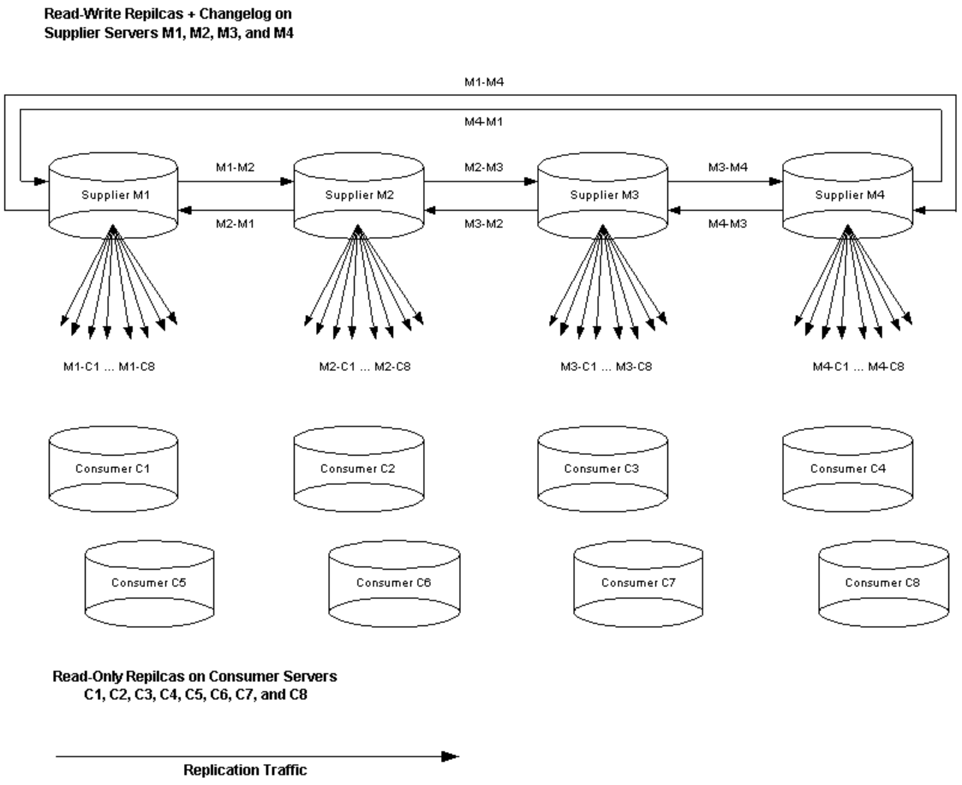

Read-write replica on Server A + change log
Supplier A
Supplier B
Read-write replica on Server B + change log
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Read-only replica on Server C
Consumer C
Consumer D
Read-only replica on Server D
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Replication traffic

Figure 8-3 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.

Figure 8-3 Multi-Master Replication (Four Suppliers)



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 Replication, especially multi-master replication, works better over high speed links than over slow links, such as a WAN, in geographically distributed environments.

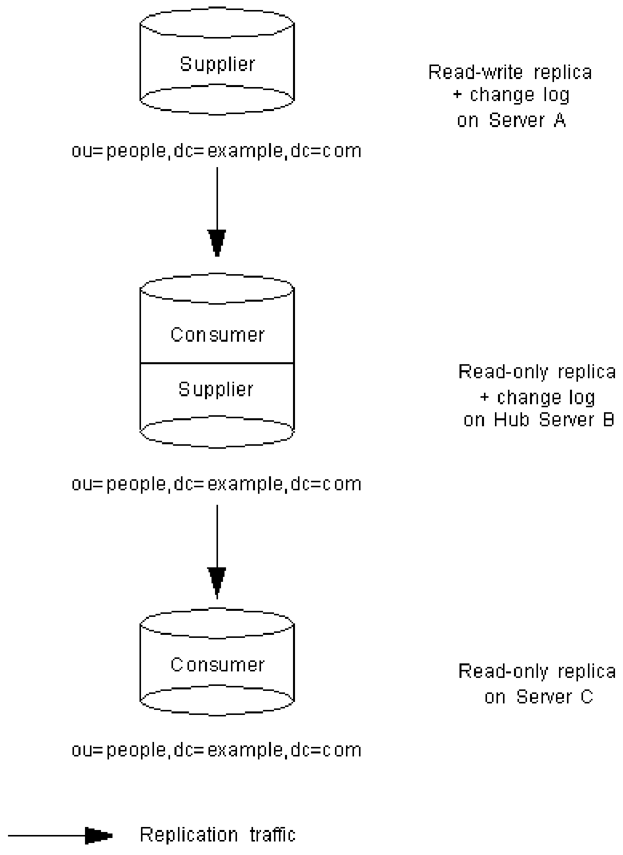
For information on setting up multi-master replication with two supplier servers and two consumer servers, refer to “Configuring Multi-Master Replication,” on page 330.

Cascading Replication

In a cascading replication scenario, one server, often called a *hub supplier*, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. 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 when you need to balance heavy traffic loads or have supplier servers based locally in geographically distributed environments.

Figure 8-4 shows an example of cascading replication. This example shows a simple cascading replication scenario. You can create more complex scenarios with several hub suppliers.

Figure 8-4 Cascading Replication


Supplier
Read-write replica + change log on Server A
ou=people,dc=example,dc=com
Consumer
Supplier
Read-only replica + change log on Hub Server B
ou=people,dc=example,dc=com
Consumer
Read-only replica on Server C
ou=people,dc=example,dc=com
Replication traffic

For information on setting up cascading replication, refer to "Configuring Cascading Replication," on page 343.

NOTE You can combine multi-master and cascading replication. For example, in the multi-master scenario illustrated in Figure 8-2, on page 314, server C and server D could be hub suppliers that would replicate to any number of consumer servers.

Handling Complex Replication Configurations

If you are configuring replication for a large number of servers and your configuration is relatively complex, for reasons of efficiency, you should proceed in the following order:

1. On all consumer servers:
    - Create the replica databases.
    - Create the Replication Manager, or supplier bind DN, entry.
    - Specify the replica settings for a read-only replica.
2. On all hub suppliers:
    - Create the replica databases.
    - Create the Replication Manager, or supplier bind DN, entry.
    - Specify the supplier settings for replication (includes changelog configuration).
    - Specify the replica settings for a read-only replica.
3. On all suppliers:
    - Create the replica databases.
    - Specify the supplier settings for replication (includes changelog configuration).
    - Specify the replica settings for a read-write replica.
4. Configure replication agreements on all suppliers:
    - Between suppliers in a multi-master set.
    - Between suppliers and dedicated consumers.

- ❍ Between suppliers and hub suppliers.

  Optionally, you can initialize the replicas on the consumer servers at this stage.

5. Configure replication agreements on all hub suppliers between the hub supplier and the dedicated consumers.

   Optionally, you can initialize the replicas on the consumer servers at this stage.

NOTE It is very important to create and configure all replicas before you attempt to create a replication agreement. This also means that when you create the replication agreement, you can choose to initialize consumers immediately.

These sections contain a description of the tasks you need to perform to configure replication:

- Creating the Supplier Bind DN Entry
- Configuring Supplier Settings
- Configuring a Read-Write Replica
- Configuring a Read-Only Replica
- Configuring a Hub Supplier
- Creating a Replication Agreement

Creating the Supplier Bind DN Entry

A critical part of setting up replication is to create the entry, referred to as the Replication Manager or supplier bind DN entry, that the suppliers will 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 supplier) 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.

For example, you could create an entry `cn=Replication Manager,cn=config` 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=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 probably suffer.

However, 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`, as this allows you to centralize configuration information.

On each server that acts as a consumer in replication agreements, create a special entry that the supplier will use to bind.

This entry must not be part of the replicated database. For example, you could use `cn=Replication Manager,cn=config`. Make sure you create the entry with the attributes required by the authentication method you specify in the replication agreement.

1. Stop the Directory Server. If you do not stop the server, the changes you make to the `dse.ldif` file will not be saved. See "Starting and Stopping the Directory Server," on page 37, 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. If you have enabled the password expiration policy or intend to do so in the future, you must remember to disable it 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.

5. The final entry should resemble this example:

```
dn: cn=replication manager,cn=config
objectClass: inetorgperson
objectClass: person
objectClass: top
cn: replication manager
sn: RM
userPassword: password
passwordExpirationTime: 20380119031407Z
```

6. Restart the Directory Server. See "Starting and Stopping the Directory Server," on page 37, for more information on starting the server.

When you configure a replica as a consumer, you must use the DN of this entry to define the supplier bind DN.

Configuring Supplier Settings

On any server that holds the master copy of a replica, you must specify supplier settings.

To configure supplier settings:

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

   For information on starting the Directory Server Console, see "Using the Directory Server Console," on page 34.

2. In the left navigation tree, highlight the Replication node.

3. In the right pane, select the Supplier Settings tab.

4. Check the Enable Changelog checkbox.

   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 Browse to display a file selector.

6. Set the changelog number and age parameters.

   You must clear the unlimited checkboxes to specify different values.

7. Click Save to save the supplier settings.

Configuring a Read-Write Replica

For each read-write replica on the supplier server, you must specify the appropriate replication settings.

To configure a read-write replica:

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

   For information on starting the Directory Server Console, see "Using the Directory Server Console," on page 34.

2. In the left navigation tree, expand the Replication folder, and highlight the database to replicate.

   The Replication tab is displayed on the right pane.

3. Check the Enable Replica checkbox.

4. In the Replica Role section, select the Single Master or Multi-Master radio button.

5. In the Common Settings section, specify a Replica ID (an integer between `1` and `254`, both inclusive).

   The replica ID must be unique for a given suffix. Make sure you specify an ID that is different from the IDs 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.

   This option indicates how often the state information stored in the replicated entries is purged.

7. Click Save to save the replication settings for the database.

Configuring a Read-Only Replica

For each read-only replica on the consumer server, you must specify the appropriate replication settings.

1. In the Directory Server Console, click the Configuration tab.

   For information on starting the Directory Server Console, see "Using the Directory Server Console," on page 34.

2. In the left navigation tree, expand the Replication folder, and then highlight the replica database.

   The Replica Settings tab is displayed on the right pane.

3. Check the Enable Replica checkbox.

4. In the Replica Role section, select the Dedicated Consumer option.

5. In the Common Settings section, specify a purge delay in the "Purge delay" field.

   This option indicates how often the state information stored in the replicated entries is purged.

6. In the Update Settings section, specify the supplier bind DN that the supplier will use to bind to the replica.

   This supplier bind DN or entry DN must correspond to the entry you created on the server that acts as a consumer in the replication agreement. You can now specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field and click Add. Your supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

   The supplier bind DN corresponds to a privileged user because it is not subject to access control.

7. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will bind over a regular connection, and, therefore, are of the form `ldap://`*hostname*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*hostname*`:`*port*, where the `s` in `ldaps` indicates secure connections.

   In the case of cascading replication, referrals are automatically sent to the hub supplier, which in turn refers the request to the original supplier. Therefore, you should set a referral to the original supplier to replace the automatically generated referral.

8. Click Save to save the replication settings for the replica.

Configuring a Hub Supplier

In a cascading replication environment, configure the hub supplier as follows:

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

   For information on starting the Directory Server Console, see "Using the Directory Server Console," on page 34.

2. In the left navigation tree, expand the Replication folder, and then highlight the database to replicate.

   The Replica Settings tab is displayed on the right pane.

3. Check the Enable Replica checkbox.

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 indicates how often the state information stored in the replicated entries is purged.

6. In the Update Settings section, specify the supplier bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. The bind DN corresponds to a privileged user because it is not subject to access control.

7. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   You can choose either to add the supplier servers that you specify to the automatically generated list or to use the supplier servers that you specify to replace the automatically generated list of servers.

8. Click Save to save the replication settings for the database.

Creating a Replication Agreement

This section explains how to create a replication agreement. You must create a replication agreement on the supplier server for each read-write replica that is supplied to a consumer server or a hub supplier.

Before you can create a replication agreement, you must have:

- Configured supplier settings on the server as described in "Configuring Supplier Settings," on page 320.
- Configured replication settings for suppliers as described in "Configuring a Read-Write Replica," on page 321.
- Configured replication settings for hub suppliers (if any) and consumers as described in "Configuring a Read-Only Replica," on page 322.

To create a replication agreement:

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

   For information on starting the Directory Server Console, see "Using the Directory Server Console," on page 34.

2. In the navigation tree, expand the Replication folder, right-click the database to replicate, and select New Replication Agreement.

   Or highlight the database, and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

   Go through the steps in the Replication Agreement Wizard by clicking Next to move to the following step.

3. In the first screen, fill in the name of your replication agreement. You may also fill in a description, such as `Supplier1 to Supplier2, 4-way MMR`.

4. On the next screen, fill in the consumer hostname and port. Unless you have more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.

   Also, select the bind method for replication. If you have enabled SSL on your servers, you may select "Using encrypted SSL connection" radio button and use SSL client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE If you have enabled attribute encryption, you must use a secure connection in order for those attributes to be replicated.

5. Optionally, you can enable fractional replication. By default, all attributes are replicated to the consumer server. If you want to select attributes that will *not* be replicated to the consumer, enable fractional replication by checking the "Enable Fractional Replication" checkbox. Then, highlight the attribute (or attributes) in the "Included" column on the right, and click "Remove." All attributes that will not be replicated will be listed in the "Excluded" column on the left, as well as in the summary when you hae finished your replication agreement.

NOTE The consumer of this replication agreement must be a dedicated consumer for fractional replication to work as a safeguard against potential data integrity problems. This is not enforced at the time you make the replication agreement, but replication will fail if your consumer is not a read-only replica. See "Replication Overview," on page 308, for more information on read-only replicas and dedicated consumers.

6. Set the replication schedule.

   By default, the servers are always kept in sync. If, for example, the connection is slow or the information being replicated doesn't change frequently, you can set a different schedule for replication, such as Monday between 12:00 p.m. and 3:00 p.m.

7. Select when to initialize the consumer.

   The default is to create an initialization file; you may also choose to initialize the consumer as soon as you finish the replication agreement or not to initialize it at all. See "Initializing Consumers," on page 350, for more information on methods and circumstances for consumer initialization.

8. The last screen is a summary of your replication agreement. Check that this is accurate, and hit "Done."

   When you have finished, an icon representing the replication agreement is displayed under the database icon. This replication agreement icon indicates that your replication agreement is set up.

NOTE Once you create a replication agreement, you cannot change the connection type (SSL or non-SSL) defined in the agreement; this is because LDAP and LDAPS connections use different ports. To change the connection type, you should re-create the replication agreement.

Configuring Single-Master Replication

This section provides information on configuring single-master replication. The steps described in this section provide a high level overview of the procedure you need to follow. Cross-references to the detailed task descriptions are provided at each step.

To set up single-master replication such as the configuration shown in Figure 8-1, on page 313, 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, you need to perform the following procedures:

- Configuring the Read-Only Replica on the Consumer Server
- Configuring the Read-Write Replica on the Supplier Server
- Initializing the Replicas for Single-Master Replication

Configuring the Read-Only Replica on the Consumer Server

1. Create the database for the read-only replica if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 82.

2. Create the entry corresponding to the supplier bind DN on the consumer server if it does not exist. This is the special entry that the supplier will use to bind to the consumer.

a. In the Directory Server Console, select the Directory tab, and create an entry. For example, you could use `cn=Replication Manager,cn=config`.

b. Specify a `userPassword` attribute-value pair.

c. If you have enabled the password expiration policy or intend to do so in future, you must remember to disable it 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.

NOTE This entry must not be part of the replicated database.

---

3. Specify the replication settings required for a read-only replica.

a. In the Directory Server Console, select the Configuration tab.

b. In the navigation tree, expand the Replication folder, and highlight the replica database.

The Replica Settings tab is displayed in the right-hand side of the window.

c. Check the Enable Replica checkbox.

d. In the Replica Role section, select the Dedicated Consumer radio button.

e. In the Common Settings section, specify a purge delay in the "Purge delay" field.

This option indicates how often the state information stored in the replicated entries is purged.

f. In the Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

You can now specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field, and click Add. You supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

g. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will bind over a regular connection and, therefore, are of the form `ldap://`*hostname*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*hostname*`:`*port*, where the `s` in `ldaps` indicates secure connections.

4. Click Save to save the replication settings for the replica.
5. Repeat these steps for every read-only replica in your replication configuration.

Configuring the Read-Write Replica on the Supplier Server

1. Specify the supplier settings for the server.
   a. In the Directory Server Console, select the Configuration tab.
   b. In the navigation tree, highlight the Replication node.
   c. In the right-hand side of the window, select the Supplier Settings tab.
   d. Check the Enable Changelog checkbox.

      This activates all of the fields in the pane below that were previously greyed out.
   e. Specify a changelog by clicking the "Use default" button, or click the Browse button to display a file selector.
   f. Set the changelog parameters (number and age).

      You must clear the unlimited checkboxes if you want to specify different values.
   g. Click Save to save the supplier settings.
2. Specify the replication settings required for a read-write replica.

a. In the navigation tree on the Configuration tab, expand the Replication node, and highlight the database to replicate.

   The Replica Settings tab is displayed in the right-hand side of the window.

b. Check the Enable Replica checkbox.

c. In the Replica Role section, select the Single Master radio button.

d. In the Common Settings section, specify a Replica ID (an integer between `1` and `254` inclusive).

   The replica ID must be unique for a given suffix, different from the IDs used for read-write replicas on this server and on other servers.

e. In the Common Settings section, specify a purge delay in the "Purge delay" field.

   This option indicates how often the state information stored in the replicated entries is purged.

f. Click Save to save the replication settings for the database.

3. Create a replication agreement.

   You must create one replication agreement for each read-only replica. For example, in the case illustrated in Figure 8-1, on page 313, server A holds two replication agreements, one for server B and one for server C.

   a. In the navigation tree of the Configuration tab, right-click the database to replicate, and select New Replication Agreement.

      Or highlight the database, and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

   b. Go through the steps in the replication wizard by clicking Next to move to the following step.

   c. When you have finished, the replication agreement is set up.

Initializing the Replicas for Single-Master Replication

You can initialize the read-only replicas from the Replication Agreement Wizard or at anytime afterwards. For information on initializing read-only replicas, refer to "Initializing Consumers," on page 350.

When you have finished, the replication agreement is set up.

Configuring Multi-Master Replication

This section provides information on 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. The steps described in this section provide a high-level overview of the procedures you need to follow for two sample multi-master replication scenarios.

- Configuring 2-Way Multi-Master Replication
- Configuring 4-Way Multi-Master Replication
- Preventing Monopolization of the Consumer in Multi-Master Replication

Configuring 2-Way Multi-Master Replication

To set up multi-master replication such as the configuration shown in Figure 8-2, on page 314, between two suppliers server A and server B, which each hold a read-write replica, and two consumers server C and server D, which each hold a read-only replica, you need to perform the following procedures:

- Configuring the Read-Only Replicas on the Consumer Servers
- Configuring the Read-Write Replicas on the Supplier Servers
- Initializing the Replicas for Multi-Master Replication

Configuring the Read-Only Replicas on the Consumer Servers

Perform these steps on each consumer server, server C and server D:

1. Create the database for the read-only replica, if it does not exist.

   For instructions, refer to “Creating Suffixes,” on page 82.

2. Create the entry corresponding to the supplier bind DN if it does not exist.

NOTE This is the special entry that the supplier will use to bind. The entry must not be part of the replicated database.

   a. In the Directory Server Console, select the Directory tab.

  b. Create an entry.

     For example you could use `cn=Replication Manager,cn=config`.

  c. Specify a `userPassword` attribute-value pair.

  d. If you have enabled the password expiration policy or intend to do so in the future, disable it for this entry to prevent replication from failing due to expiration of passwords.

     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.

3. Specify the replication settings required for a read-only replica.

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, expand the Replication folder, and then select the replica database.

      The Replica Settings tab is displayed on the right pane.

   c. Check the Enable Replica checkbox.

   d. In the Replica Role section, select the Dedicated Consumer radio button.

   e. In the Common Settings section, specify a purge delay in the "Purge delay" field.

      This option indicates how often the state information stored in the replicated entries is purged.

   f. In the Update Settings section, specify the bind DN or entry DN that the supplier will use to bind to the replica.

      This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

      You can specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field, and click Add. You supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

g. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will perform a simple bind and, therefore, are of the form `ldap://`*hostname*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*hostname*`:`*port*, where the `s` in `ldaps` indicates secure connections.

4. Click Save to save the replication settings for the replica.
5. Repeat these steps for every read-only replica in your replication configuration.

Configuring the Read-Write Replicas on the Supplier Servers

Perform these steps on each supplier server, server A and server B:

1. Specify the supplier settings for each server.

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, highlight the Replication node.

   c. In the right-hand side of the window, select the Supplier Settings tab.

   d. Check the Enable Changelog checkbox.

      This activates all of the fields in the pane below.

   e. Specify a changelog by clicking the "Use default" button, or click the Browse button to display a file selector.

   f. Set the changelog parameters (number and age).

      You must clear the unlimited checkboxes if you want to specify different values.

   g. Click Save to save the supplier settings.

2. Create the entry corresponding to the supplier bind DN if it does not exist.

   For multi-master replication, it is necessary to create this supplier bind DN on the supplier servers (as well as the consumers) because they act as both consumer and supplier to the other supplier servers.

NOTE This is the special entry that the supplier will use to bind. The entry must not be part of the replicated database.

a. In the Directory Server Console, select the Directory tab.

b. Create an entry.

   For example, you could use `cn=Replication Manager,cn=config`.

c. Specify a `userPassword` attribute-value pair.

d. If you have enabled the password expiration policy or intend to do so in the future, disable it to prevent replication from failing due to expiration of passwords.

   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.

3. On server A and server B, specify the replication settings for the multi-mastered read-write replica.

   a. In the Directory Server Console, select the Configuration tab.

   b. Expand the Replication node, and select the database to replicate.

      The Replica Settings tab is displayed on the right pane.

   c. Check the Enable Replica checkbox.

   d. In the Replica Role section, select the Multiple Master radio button.

   e. In the Common Settings section, specify a Replica ID.

      The replica ID must be an integer between `1` and `254`, both inclusive, and must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

   f. In the Common Settings section, specify a purge delay in the "Purge delay" field.

      This option indicates how often the state information stored in the replicated entries is purged.

g. In the Update Settings section, specify the supplier bind DN or entry DN that the supplier will use to bind to the replica.

   This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

h. Specify the supplier server to which you want to refer updates (the other supplier in the multi-master set).

   Only specify the URL for the supplier server if you want clients to bind using SSL. In such a case, you must specify a URL beginning with `ldaps://`.

i. Click Save to save the replication settings for the database.

4. On server A, set up the following replication agreements:

   - One with supplier server B, where server B is configured as a *consumer* for the replica.
   - One for each consumer servers, server C and server D.

   To do this:

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, right-click the database to replicate, and select New Replication Agreement.

      Or highlight the database, and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

   c. Go through the steps in the replication wizard by clicking Next to move to the following step.

      You can initialize the read-only replicas and the read-write replica on server B from the Replication Agreement Wizard or at anytime afterwards. For information on the order and procedure for initializing read-only replicas, refer to “Initializing the Replicas for Multi-Master Replication,” on page 335, and “Initializing Consumers,” on page 350.

      When you have finished, the replication agreement is set up.

5. On server B, set up the following replication agreements:

   - One with supplier server A, where server A is declared as a *consumer* for the replica. During this operation, do not initialize server A from server B if you have already initialized server B from server A in Step 4.

- ❍ One for each consumer servers, server C and server D.

NOTE Once you have completed these procedures, server A and server B have mutual replication agreements, which means that they can accept updates from each other.

When you have configured the servers holding the read-write replicas, the necessary replication agreements, and the servers holding the read-only replicas, you are ready to initialize replication. You can perform this task when you create the replication agreements on the supplier servers or at any time afterwards.

Initializing the Replicas for Multi-Master Replication

In the case of multi-master replication, you should initialize replicas in the following order:

1. Ensure one supplier has the complete set of data to replicate. Use this supplier to initialize the replicas on the other masters in the multi-master replication set.
2. Initialize the replicas on the consumer servers from any one of the two suppliers.

For information on initializing replicas, refer to “Initializing Consumers,” on page 350.

Configuring 4-Way Multi-Master Replication

Directory Server supports 4-way multi-master replication. To set up multi-master replication such as the configuration shown in Figure 8-3, on page 315, between four supplier servers, server M1 through server M4, that each hold a read-write replica, and eight consumer servers, server C1 through server C8, that each hold a read-only replica, you need to perform the following procedures:

- Configuring the Read-Only Replicas on the Consumer Servers
- Configuring the Read-Write Replicas on the Supplier Servers
- Initializing the Replicas for Multi-Master Replication

NOTE If you have more than 10 databases running with replication, you may see performance degradation. Also, if you have more than 20 replication agreements on a supplier, you may see performance degradation. If you need to support that many consumers, you may have to introduce one or more hub replicas between your supplier(s) and consumers. See "Configuring Cascading Replication," on page 343.

Configuring the Read-Only Replicas on the Consumer Servers

Perform these steps on each consumer server, server C1 through server C8:

1. Create the database for the read-only replica if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 82.

2. Create the entry corresponding to the supplier bind DN if it does not exist.

NOTE This is the special entry that the supplier will use to bind. The entry must not be part of the replicated database.

---

   a. In the Directory Server Console, select the Directory tab.

   b. Create an entry.

      For example, you could use `cn=Replication Manager,cn=config`.

   c. Specify a `userPassword` attribute-value pair.

   d. If you have enabled the password expiration policy or intend to do so in the future, disable it to prevent replication from failing due to expiration of passwords.

      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.

3. Specify the replication settings required for a read-only replica.

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, expand the Replication folder, and select the replica database.

      The Replica Settings tab is displayed on the right pane.

c. Check the Enable Replica checkbox.

d. In the Replica Role section, select the Dedicated Consumer radio button.

e. In the Common Settings section, specify a purge delay in the "Purge delay" field.

   This option indicates how often the state information stored in the replicated entries is purged.

f. In the Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

   You can specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field, and click Add. You supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

g. Specify the supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will perform a simple bind and, therefore, are of the form `ldap://`*hostname*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*hostname*`:`*port*, where the `s` in `ldaps` indicates secure connections.

4. Click Save to save the replication settings for the replica.

5. Repeat these steps for every read-only replica in your replication configuration.

Configuring the Read-Write Replicas on the Supplier Servers

Perform these steps on each supplier server, server M1 through server M4:

1. Specify the supplier settings for each server.

   a. In the Directory Server Console, select the Configuration tab.

b. In the navigation tree, highlight the Replication node, and, on the right pane, select the Supplier Settings tab.

c. Check the Enable Changelog checkbox.

   This activates all of the fields in the pane below.

d. Specify a changelog by clicking the Use Default button, or click the Browse button to display a file selector.

e. Set the changelog parameters (number and age).

   You must clear the unlimited checkboxes if you want to specify different values.

f. Click Save to save the supplier settings.

2. Create the entry corresponding to the supplier bind DN if it does not already exist.

   For multi-master replication, it is necessary to create this supplier bind DN on the supplier servers (as well as the consumers) because they act as both consumer and supplier to the other supplier servers.

NOTE This is the special entry that the supplier will use to bind. The entry must not be part of the replicated database.

a. In the Directory Server Console, select the Directory tab.

b. Create an entry.

   For example, you could use `cn=Replication Manager,cn=config`.

c. Specify a `userPassword` attribute-value pair.

d. If you have enabled the password expiration policy or intend to do so in the future, disable it to prevent replication from failing due to expiration of passwords.

   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.

3. Specify the replication settings for the multi-mastered read-write replica.

a. In the navigation tree of the Configuration tab, expand the Replication node, and then select the database to replicate.

   The Replica Settings tab is displayed on the right pane.

b. Check the Enable Replica checkbox.

c. In the Replica Role section, select the Multiple Master radio button.

d. In the Common Settings section, specify a Replica ID.

   The replica ID must be an integer between `1` and `254`, both inclusive, and must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

e. In the Common Settings section, specify a purge delay in the “Purge delay” field.

   This option indicates how often the state information stored in the replicated entries is purged.

f. In the Update Settings section, specify the supplier bind DN or entry DN that the supplier will use to bind to the replica.

   This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

g. Specify the supplier server to which you want to refer updates (the other suppliers in the multi-master set).

   Only specify the URL for the supplier server. If you want clients to bind using SSL, you must specify a URL beginning with `ldaps://`.

h. Click Save to save the replication settings for the database.

4. Set up replication agreements on all the supplier servers:

   ❍ On server M1, set up the following replication agreements: one with supplier server M2, where server M2 is configured as a *consumer* for the replica; one with supplier server M4, where server M4 is configured as a *consumer* for the replica; and one for each consumer servers, server C1 through server C8.

- On server M2, set up the following replication agreements: one with supplier server M1, where server M1 is declared as a *consumer* for the replica; one with supplier server M3, where server M3 is declared as a *consumer* for the replica; and one for each consumer servers, server C1 through server C8.
- On server M3, set up the following replication agreements: one with supplier server M2, where server M2 is declared as a *consumer* for the replica; one with supplier server M4, where server M4 is declared as a *consumer* for the replica; and one for each consumer, server C1 through server C8.
- On server M4, set up the following replication agreements: one with supplier server M3, where server M3 is declared as a *consumer* for the replica; one with supplier server M1, where server M1 is declared as a *consumer* for the replica; and one for each consumer servers, server C1 through server C8.

To set up a replication agreement:

a. In the Directory Server Console, select the Configuration tab.

b. In the navigation tree, right-click the database to replicate, and then select New Replication Agreement.

Or highlight the database, and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

c. Go through the steps in the replication wizard by clicking Next to move to the following step.

When you have finished, the replication agreement is set up.

NOTE Once you have completed these procedures, all four servers — server M1 through server M4 — have mutual replication agreements, which means that they can accept updates from each other.

When you have configured the servers holding the read-write replicas, the necessary replication agreements, and the servers holding the read-only replicas, you are ready to initialize replication. You can perform this task when you create the replication agreements on the supplier servers or at any time afterwards. For information on the order and procedure for initializing read-only replicas, refer to "Initializing the Replicas for Multi-Master Replication," on page 335, and "Initializing Consumers," on page 350.

During this operation, do not try to reinitialize the servers. For example, do not initialize server M1 from server M2 if you have already initialized server M2 from server M1.

Initializing the Replicas for Multi-Master Replication

In the case of multi-master replication, you should initialize replicas in the following order:

1. Ensure one supplier has the complete set of data to replicate. Use this supplier to initialize the replica on the other supplier in the multi-master replication set.
2. Initialize the replicas on the consumer servers from either of the four suppliers.

For information on initializing replicas, refer to “Initializing Consumers,” on page 350.

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.

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`

  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`

  Amount of time in seconds a supplier should wait between update sessions. Set this interval so that it is at least 1 second longer than the interval specified for `nsds5ReplicaBusyWaitTime`. Increase the interval as needed until you reach 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. You can 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 you set either attribute 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 1 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 no longer logged by default because they are usually benign. If you want to see them, turn on the replication error log level.

Configuring Cascading Replication

This section provides information on setting up cascading replication. The steps described in this section provide a high-level overview of the procedures you need to follow, and cross-references to the detailed task descriptions are provided at each step.

To set up cascading replication such as the configuration shown in Figure 8-4, on page 316, between 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, you need to perform the following procedures:

- Configuring the Read-Only Replica on the Consumer Server
- Configuring the Read-Only Replica on the Hub Supplier
- Configuring the Read-Write Replica on the Supplier Server
- Initializing the Replicas for Cascading Replication

Configuring the Read-Only Replica on the Consumer Server

To configure the read-only replica in a consumer server:

1. On the consumer server, create the database for the replica if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 82.

2. On the consumer server, create the entry corresponding to the supplier bind DN if it does not exist. This is the special entry that the supplier will use to bind.

3. On the consumer server, specify the replication settings for the read-only replica.

   a. In the Directory Server Console, select the Configuration tab.

   b. In the navigation tree, expand the Replication folder, and highlight the replica database.

      The Replica Settings tab is displayed on the right pane.

   c. Check the Enable Replica checkbox.

   d. In the Replica Role section, select the Dedicated Consumer radio button.

- **e.** In the Common Settings section, specify a purge delay in the "Purge delay" field.

  This option indicates how often the state information stored in the replicated entries is purged.

- **f.** In the Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

  This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

  You can specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field, and click Add. You supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

- **g.** Specify any supplier servers to which you want to refer updates.

  By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

  In the case of cascading replication, referrals are automatically sent to the hub supplier, which in turn refers the request to the original supplier. Therefore, you should set a referral to the original supplier to replace the automatically generated referral.

4. On the supplier server, set up the replication agreement between this server and the hub supplier.

- **a.** In the navigation tree of the Configuration tab, right-click the database to replicate, and then select New Replication Agreement.

  Or highlight the database, and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

- **b.** Go through the steps in the replication wizard by clicking Next to move to the following step.

  You can initialize the read-only replicas from the Replication Agreement Wizard or at anytime afterwards. For information on initializing read-only replicas, refer to "Initializing Consumers," on page 350.

5. On the hub supplier, set up the replication agreement between this server and the consumer.

When you have configured the replicas on each server and the necessary replication agreements among servers, you can initialize the read-only replicas on the hub supplier and on the consumer. You can perform this task from the Replication Agreement Wizard while you are configuring the supplier server and the hub supplier server or at any time afterwards.

Configuring the Read-Only Replica on the Hub Supplier

Perform these steps on the hub supplier that receives replication updates from the supplier and propagates them to consumers:

1. Create the database for the replica if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 82.

2. Create the entry corresponding to the supplier bind DN if it does not exist.

NOTE	This is the special entry that the supplier will use to bind. The entry must not be part of the replicated database.

   a. In the Directory Server Console, select the Directory tab.

   b. Create an entry.

      For example you could use `cn=Replication Manager,cn=config`.

   c. Specify a `userPassword` attribute-value pair.

   d. If you have enabled the password expiration policy or intend to do so in the future, disable it to prevent replication from failing due to expiration of passwords.

      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.

3. Specify the required replication settings.

   a. In the Directory Server Console, select the Configuration tab.

b. In the navigation tree, expand the Replication node, and then select the database to replicate.

The Replica Settings tab is displayed on the right pane.

c. Check the Enable Replica checkbox.

d. In the Replica Role section, select the Hub radio button.

e. In the Common Settings section, specify a purge delay in the "Purge delay" field.

This option indicates how often the state information stored in the replicated entries is purged.

f. In the Update Settings section, specify the supplier bind DN that the supplier will use to bind to the replica.

This supplier bind DN should correspond to the entry created in Step 2. The supplier bind DN corresponds to a privileged user because it is not subject to access control.

You can specify multiple supplier bind DNs per replica but only one supplier DN per replication agreement. To specify your supplier bind DN, enter your supplier bind DN in the "Enter a new Supplier DN" field, and click Add. Your supplier bind DN will appear in the Current Supplier DNs list. Repeat the operation for every supplier bind DN you want to include in the list.

g. Specify any supplier servers to which you want to refer updates.

By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients will bind over a regular connection and, therefore, are of the form `ldap://`*hostname*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*hostname*`:`*port*.

h. Click Save to save the replication settings for the database.

Configuring the Read-Write Replica on the Supplier Server

Perform these steps on the supplier server that holds the original copy of the database:

1. Specify the supplier settings for the server.

    a. In the Directory Server Console, select the Configuration tab.

    b. In the navigation tree, highlight the Replication node.

    c. On the right pane, select the Supplier Settings tab.

    d. Check the Enable Changelog checkbox.

    This activates all of the fields in the pane.

    e. Specify a changelog by clicking the "Use default" button, or click the Browse button to display a file selector.

    f. Set the changelog parameters (number and age).

    You must clear the unlimited checkboxes if you want to specify different values.

    g. Click Save to save the supplier settings.

2. Specify the required replication settings.

    a. In the navigation tree of the Configuration tab, expand the Replication node, and then highlight the database to replicate.

    The Replica Settings tab is displayed in the right-hand side of the window.

    b. Check the Enable Replica checkbox.

    c. In the Replica Role section, select the Single Master radio button.

    d. In the Common Settings section, specify a Replica ID.

    The replica ID must be an integer between `1` and `254`, both inclusive, and must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

e. In the Common Settings section, specify a purge delay in the “Purge delay” field.

   This option indicates how often the state information stored in the replicated entries is purged.

f. Click Save to save the replication settings for the database.

Initializing the Replicas for Cascading Replication

In the case of cascading replication, you should initialize replicas in the following order:

1. Use the supplier server to initialize the replica on the hub supplier.
2. From the hub supplier, initialize the replica on the consumer.

For information on initializing replicas, refer to “Initializing Consumers,” on page 350.

Making a Replica Updatable

To make a read-only server writable, follow this procedure:

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, and then, on the right pane, enable changelog.
5. Select the suffix, and, in the Replica Settings tab, change Replica Role to Single Master, and assign a unique replica ID.
6. Save your changes, and restart the server.

Deleting 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). In the event of a supplier server going 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. Once you have deleted the changelog, you can initialize your consumers and begin the replication process afresh. To delete the changelog, you can either remove it or move it to a new location.

This section contains the information for the following procedures:

- Removing the Changelog
- Moving the Changelog to a New Location

Removing the Changelog

You can remove the changelog using the Directory Server Console. To remove the changelog from the supplier server:

1. In the Directory Server Console, select the Configuration tab.
2. Select the Replication Agreements folder in the left navigation tree and then the Supplier Server Settings tab in the right pane.
3. Clear the Enable Changelog checkbox.

   This deletes the changelog.
4. Click Save.
5. Restart the Directory Server.
6. Reinitialize your consumers.

   See "Initializing Consumers," on page 350, for information.

NOTE If you remove the changelog, you will need to reinitialize your consumer servers.

Moving the Changelog to a New Location

To delete the changelog while the server is still running and continuing to log changes, you simply move the changelog to a new location. By moving the changelog, a new changelog is created in the directory you specify, and the old changelog is deleted. If you change the location of the changelog, it is as if you are reinitializing it, which in turn requires consumer reinitialization.

For example, you could move the changelog from the default location of

serverRoot`/slapd-`*serverID*`/changelogdb`

to

serverRoot`/slapd-`*serverID*`/newchangelogdb`

Initializing Consumers

Once you have created a replication agreement, you must initialize the consumer; that is, you must physically copy data from the supplier server to the consumer servers. This section first describes consumer initialization in detail and then provides instructions on the two different methods for initializing consumers. This section is divided into the following parts:

- When to Initialize a Consumer
- Online Consumer Initialization Using the Console
- Manual Consumer Initialization Using the Command-Line
- Filesystem Replica Initialization

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 initialized again. However, if the data on the supplier server is restored from backup for any reason, then you should reinitialize all of the consumers supplied by it.

You can either initialize the consumer 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 you initialize the consumer while 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.

Online Consumer Initialization Using the Console

Online consumer initialization using the Console is the easiest way to initialize or reinitialize a consumer. However, if you are replicating across a slow link, this process can be very time-consuming, and you may find manual consumer initialization using the command-line to be a more efficient approach. Refer to "Manual Consumer Initialization Using the Command-Line," on page 352, for more information.

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.

Performing Online Consumer Initialization

To initialize or reinitialize a consumer online:

1. Create a replication agreement.

   See "Creating a Replication Agreement," on page 324.

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 is displayed to warn you 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. You can check the status of the online consumer initialization on the Summary tab in the Status box. If online consumer initialization is in progress, the status shows that a replica is being initialized.

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 "Monitoring Replication Status," on page 366.

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. We suggest you use the manual process whenever you find that the online process is inappropriate due to performance concerns.

This section is divided into the following parts:

- Manual Consumer Initialization Overview
- Exporting a Replica to LDIF
- Importing the LDIF File to the Consumer Server

Manual Consumer Initialization Overview

To initialize or reinitialize a server manually:

1. Create a replication agreement.

   See "Creating a Replication Agreement," on page 324.

2. Export the replica on the supplier server to an LDIF file.

   See "Exporting a Replica to LDIF," on page 353.

3. Import the LDIF file with the supplier replica contents to the consumer server.

   See "Importing the LDIF File to the Consumer Server," on page 353, for instructions.

Exporting a Replica to LDIF

You can convert the replica to LDIF using one of the following three procedures:

1. When you create a replication agreement by selecting "Create consumer initialization file" in the Initialize Consumer dialog box of the Replication Agreement Wizard.
2. From the Directory Server Console at any time by right-clicking the replication agreement under the Replication folder and choosing "Create LDIF File" from the pop-up menu.
3. From the command-line by using the export command as described in "Exporting to LDIF from the Command-Line," on page 161.

Importing the LDIF File to the Consumer Server

You can 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 "Importing from the Command-Line," on page 155.

If you use the `ldif2db` script, remember to bind using the supplier bind DN configured on the consumer server.

NOTE If you use the `ldif2db.pl` script, the LDIF file import operation does not require a server restart. For more information on command-line scripts, see *Red Hat Directory Server Configuration, Command, and File Reference*.

Filesystem Replica Initialization

If you have a very large database, such as one with several million entries, it can take an hour or more to initialize a consumer from the Console or even with manual initialization. To save time, you can 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 via 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, it is recommended that you still use the 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, Sun Solaris 32-bit to Sun Solaris 32-bit, HP-UX 64-bit to HP-UX 64-bit.

Initializing the Consumer Replica from the Backup Files

1. Create a new database on the destination server to match the database from the source server.

   Before you begin initializing the consumer from the backup files, be certain that you have created the appropriate database on your destination server so that the database exists to be "restored" and initialized.

2. Enable replication on the backend as a dedicated consumer.

3. If you already have a replication agreement to that host and port, than replication should resume immediately after running the restore script. Otherwise, create the replication agreement on your source server (or whatever server you will use as the supplier), and select the "Do not initialize consumers at this time" radio button.

4. At the command prompt, change to the following directory on the source Directory Server:

   `cd` *serverRoot*`/slapd-`*serverID*

5. Stop the source Directory Server if it is running by typing the following:

   ```
   ./stop-slapd
   ```

6. From the command-line, run the `db2bak` utility, and archive your current directory installation. You can also create a new backup by hitting the "Back Up Directory Server" button in the Tasks tab of the Directory Server Console and then putting the name of your archive directory in the "Directory:..." field.

   You can select any previous back-up to initialize the consumer if you prefer.

   This information is recovered in the destination replica.

7. Restart the source Directory Server by typing the following:

   ```
   ./start-slapd
   ```

8. Copy the archived files to a directory on your destination server, such as `archiveDirectory`.

9. At the command prompt, change to the following directory on the destination Directory Server:

   ```
   cd serverRoot/slapd-serverID
   ```

10. Stop the destination Directory Server if it is running by typing the following:

    ```
    ./stop-slapd
    ```

11. On the destination server, restore the archives with the `bak2db` script, using the optional `-n` parameter to specify the backend instance name. This `-n` parameter is similar to the `-n` used with `ldif2db` and `db2ldif`. For example:

    ```
    ./bak2db /redhat/servers/slapd-serverID/archiveDirectory -n
    userRoot
    ```

12. Restart the destination Directory Server by typing the following:

    ```
    ./start-slapd
    ```

Replication will begin on schedule as soon as the destination server is restarted.

For more information on using these scripts, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Forcing Replication Updates

When you stop a Directory Server involved in replication for regular maintenance, when it comes back online, you need to ensure that it gets updated through replication immediately. 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 supplier or a dedicated consumer is taken offline for maintenance, when they come back online, they need to be updated by the supplier server.

If you have configured replication agreements to keep the supplier server and the consumer server in sync always, this is not sufficient to bring back up-to-date a server that has been offline for over five minutes. The reason is that, with the "Always Keep in Sync" option, 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, you can use either the Directory Server Console on the supplier server that holds the reference copy of the directory information or a customizable script.

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, you can perform these steps 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 the database nodes until you select the replication agreement corresponding to the replica that you must 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.

Forcing Replication Updates from the Command-Line

From the consumer that requires updating, you can run a script that prompts the supplier to send replication updates immediately. This script is shown in Code Example 8-1.

You can copy this example and give it a meaningful name, such as `replicate_now.sh`. You must provide actual values for the variables listed in Code Example 8-1.

NOTE You must run this script since it cannot be configured to run automatically as soon as the server, which was offline, comes back online again.

Code Example 8-1 Replicate_Now 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 -1 -T -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}))" dn nsds5ReplicaUpdateSchedule >
/tmp/$$

cat /tmp/$$ |
awk '
BEGIN { s = 0 }
/^dn: / { print $0;
    print "changetype: modify";
    print "replace: nsds5ReplicaUpdateSchedule";
    print "nsds5ReplicaUpdateSchedule: 0000-2359 0123456";
    print "-";
    print "";
    print $0;
    print "changetype: modify";
    print "replace: nsds5ReplicaUpdateSchedule";
}
```

Code Example 8-1 Replicate_Now Script Example *(Continued)*

```
/^nsds5ReplicaUpdateSchedule: / { s = 1; print $0; }

/^$/ {
  if ( $s == 1 )
    { print "-" ; print ""; }
  else
    { print "nsds5ReplicaUpdateSchedule: 0000-2359 0123456";
      print "-" ; print ""; };
  s = 0; }

' > /tmp/ldif.$$

echo "Ldif is in /tmp/ldif.$$"
echo

ldapmodify -c -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \
-w ${SUP_MGRPW} -f /tmp/ldif.$$
```

If you intend to use this script, you must replace the variables with actual values in your replication environment.

Table 8-1 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.

If you want the update operation to occur over an SSL connection, you must modify the `ldapmodify` command in the script with the appropriate parameters and values. For more information on the `ldapmodify` command, refer to “Managing Entries from the Command-Line,” on page 57, and *Red Hat Directory Server Configuration, Command, and File Reference*.

Replication over SSL

You can configure Directory Servers involved in replication so that all replication operations occur over an SSL connection.

To use replication over SSL, you must first do the following:

- Configure both your supplier and consumer servers to use SSL.
- Configure your consumer server to recognize your supplier server’s certificate as the supplier DN. You do this only if you want to use SSL client authentication rather than simple authentication.

These procedures are described in chapter 11, “Managing SSL and SASL.”

NOTE Replication configured over SSL with certificate-based authentication will fail in the following cases:

- If the supplier’s certificate is a self-signed certificate.
- If the supplier’s certificate is only capable of behaving as an SSL server certificate, meaning it is unable to play the role of the client during an SSL handshake.

When your servers are configured to use SSL, you can ensure replication operations occur over SSL connections by using the Replication Agreement Wizard, which enables you to set up a replication agreement between two Directory Servers. Keep in mind that once you create a replication agreement, you cannot change the connection type (SSL or nonSSL) defined in the agreement; this is because LDAP and LDAPS connections use different ports. To change the connection type, you must re-create the replication agreement.

NOTE If you have enabled attribute encryption, you must use a secure connection for replication.

Configuring Replication over SSL Using the Replication Agreement Wizard

1. In the Directory Server Console of the supplier server, click the Configuration tab, expand the Replication folder, and select the database that you want to replicate.
2. Right-click the database, and choose New Replication Agreement from the drop-down menu.

   The Replication Agreement Wizard is displayed.
3. Go through each step in the Replication Agreement Wizard until you reach the Source and Destination window.
4. In the Connection section, check "Using Encrypted SSL Connection."
5. Select "SSL Client Authentication" or "Simple Authentication."

   If you select SSL Client Authentication, the supplier and consumer servers will use certificates to authenticate to each other.

   If you select Simple Authentication, the supplier and consumer servers will use a bind DN and password to authenticate to each other. You must specify this information in the text fields provided. When you specify this option, simple authentication takes place over a secure channel but without certificates.
6. Click Next, and proceed with the replication setup.

Replication with Earlier Releases

This section provides information on how to optimize replication with earlier releases of Directory Server. This version of Directory Server can be involved in replication scenarios with earlier releases of Directory Server, providing the following conditions are met:

- Directory Server is defined as a consumer in the replication agreement.
- The legacy suppliers can be Directory Server 4.0, 4.1, and 4.1x.

The following restrictions apply:

- A legacy Directory Server and this version of Directory Server 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 this version of Directory Server.

- This version of Directory Server cannot be a supplier for other replicas.

The main advantage of being able to use this version of Directory Server as a consumer of a legacy Directory Server is to ease the migration of a replicated environment. For more information on the steps to follow to migrate a replicated environment, refer to the *Red Hat Directory Server Installation Guide*.

Configuring Directory Server as a Consumer of a Legacy Directory Server

If you intend to use your Directory Server as a consumer of an earlier release of Directory Server, you must configure it as follows:

1. In the Directory Server Console, click the Configuration tab.

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 34.

2. In the Configuration tab, select the Replication node, and click the Legacy Consumer Settings tab in the right pane.

3. Check the Enable Legacy Consumer checkbox.

   This activates the fields in the Authentication box.

4. Specify the Supplier DN that the legacy supplier server will use to bind.

   Optionally, you can specify a password. The password must contain at least 8 characters.

5. Click Save.

   You must now configure legacy consumer settings for each replica that will receive updates from a legacy supplier.

6. In the navigation tree, expand the Replication node, and select a replica that will receive updates from the legacy supplier.

7. In the Replica Settings tab, select the Enable Replica and "Updatable by a 4.x Replica" checkboxes.

   These options are the only ones required for replication to work. You can optionally specify a Replica ID. It is not necessary to specify a Supplier DN because the one you specified in Step 4 will be used.

8. Click Save.

9. Repeat Step 7 and Step 8 for each read-only replica that will receive updates from a legacy supplier.

10. To complete your legacy replication setup, you must now configure the legacy supplier to replicate to the Directory Server. For instructions on configuring a replication agreement on a 4.x Directory Server, refer to the documentation for your 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 you can configure your Directory Server as you want it to be after the migration and activate legacy consumer settings just for the duration of the transition.

Using the Retro Changelog Plug-in

The retro changelog plug-in enables you to configure 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 in deployments where Directory Server coexists with the retired Netscape Meta Directory application. You might also need to maintain a retro changelog if you have directory clients that depend on a Directory Server 4.x style changelog.

To use the retro changelog plug-in, Directory Server must be configured as a supplier server in a single-master replication scenario.

When you have configured Directory Server 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 8-2.

Table 8-2 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.

Table 8-2 Attributes of a Retro Changelog Entry *(Continued)*

Attribute	**Definition**
`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 one of the following values: `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:

- Enabling the Retro Changelog Plug-in
- Trimming the Retro Changelog
- Searching and Modifying the Retro Changelog
- Retro Changelog and the Access Control Policy

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
   cn: Retro Changelog Plugin
   changetype: modify
   replace: nsslapd-pluginenabled
   nsslapd-pluginenabled: on
   ```

2. Use the `ldapmodify` command to import the LDIF file into the directory.

   For more information on the `ldapmodify` command, refer to "Managing Entries from the Command-Line," on page 57, and *Red Hat Directory Server Configuration, Command, and File Reference*.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

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, refer to "Enabling and Disabling Plug-ins from the Server Console," on page 516.

Trimming the Retro Changelog

The entries in the changelog can be automatically removed after a specified period of time. To configure the period of time after which entries are automatically deleted from the changelog, you must set the `nsslapd-changelogmaxage` configuration attribute in the `cn=Retro Changelog Plugin,cn=plugins,cn=config` entry.

The `nsslapd-changelogmaxage` attribute is a single-valued attribute. Its syntax is as follows:

`nsslapd-changelogmaxage:` *Integer timeUnit*

where *Integer* represents a number and *timeUnit* can be one of the following: `s` for seconds, `m` for minutes, `h` for hours, `d` for days, or `w` for weeks.

There should not be a space between the *Integer* and *timeUnit* variables. The space in the syntax above is intended to show that the attribute value is composed of two variable parts, not just one.

Example of `nsslapd-changelogmaxage` value:

```
nsslapd-changelogmaxage: 2d
```

Searching and Modifying the Retro Changelog

The changelog supports search operations. It is optimized for searches that include filters of the form:

```
(&(changeNumber>=X)(changeNumber<=Y))
```

As a general rule, you should not perform add or modify operations on the retro changelog entries, although you can delete entries to trim the size of the changelog. You will only need to perform a modify operation on the retro changelog is to modify the default access control policy.

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.

You should 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, you can modify the `aci` attribute of the `cn=changelog` entry.

Monitoring Replication Status

You can monitor replication status using the Directory Server Console and Red Hat Administration Express. This section explains both these procedures:

- Monitoring Replication Status from the Directory Server Console

- Monitoring Replication Status from Administration Express

Monitoring Replication Status from the Directory Server Console

To view a summary of replication status via the Directory Server Console:

1. Open the Directory Server Console.
2. 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.
3. Click Refresh to update the contents of the tab.

   The status information displayed is described in Table 8-3.

Table 8-3 Directory Server Console - Replication Status

Table Header	Description
Agreement	Contains the name you provided when you set up the replication agreement.
Replica suffix	Contains the suffix that is replicated.
Supplier	Specifies the supplier server in the agreement.
Consumer	Specifies the consumer server in the agreement.
Number of changes	Indicates the number of changes sent to this replica since the server started.
Last replica update began	Indicates when the most recent replication update started.
Last replica update ended	Indicates when the most recent replication update ended.
Last update message	Provides the status for the most recent replication updates.
Consumer initialization	Provides current status on consumer initialization (in progress or not).
Last consumer initialization update message	Provides status on the last initialization of the consumer.
Last consumer initialization began	States when the initialization of the consumer replica started.
Last consumer initialization ended	States when the initialization of the consumer replica ended.

Monitoring Replication Status from Administration Express

Although the replication status report that you view via the Directory Server Console shows many details, it does not show the progress of the replication. Additionally, because one report is generated per agreement, you need to navigate among the status reports for different agreements.

The `template-repl-monitor.pl` script, which is explained in detail in the *Red Hat Directory Server Configuration, Command, and File Reference*, enables you to monitor replication status to a greater extent by providing these functionalities:

- Lists for each supplier replica on each Directory Server discovered, server URL or alias, replica ID, replica root, and maximum change sequence number (`maxcsn`).
- Lists corresponding to each supplier replica listed above and for each direct or indirect consumer replicas discovered, server URL or alias, replica root, replica type, connection type of the replication sessions, replication schedule, replication status, supplier `maxcsn`, and time lag between the consumer `maxcsn` and the supplier `maxcsn`.

  The time lag field uses different colors to indicate the degree of time lag. The threshold for each color is configurable.
- Displays the change sequence number (CSN) in human-readable format (instead of hex strings) in the `MM/DD/YYYY HH:MI Seq# SubSeq#` format, where `Seq#` and `SubSeq#` are omitted if they are zero.
- Shows the output/result in the HTML format. The script writes the output to an HTML file, which can be configured to refresh itself automatically; the refresh interval is also configurable.

The script is integrated into the Red Hat Administration Express, enabling you to view the replication status via an HTTP interface. (The Administration Express is an HTML-based version of Red Hat Console that provides quick access to servers running Administration Server.)

To view in-progress status of replication via the Administration Express interface:

1. Prepare a configuration file following the guidelines provided in the "template-repl-monitor.pl (Monitor replication status)" section of the *Red Hat Directory Server Configuration, Command, and File Reference*.
2. Open a web browser window.

3. In the URL field, enter the Administration Server URL in this format:

   `http://`*hostname*`:`*admin_port*

4. Click Red Hat Administration Express, and, when prompted, log in.

5. Select a supplier Directory Server instance, and click Replication Status.

   This brings up a page for specifying the runtime parameters of the replication-monitoring tool.

6. In the "Configuration file" field, type the path to the configuration file you created in Step 1, and click OK.

   The replication-status page appears; by default, the page gets refreshed every `300` seconds.

   Each table shows the status of the changes originated from a supplier replica.

   - **Table Header —** The table header shows the replica ID of the supplier replica, the replica root, and the maximum Change State Number (CSN) on the supplier. The important thing is to make sure that each supplier LDAP server has its unique replica ID. Multiple replica roots on one LDAP server, however, could share the same replica ID.
   - **Table Row —** Each row represents a direct or indirect consumer of the supplier (identified in the Table Header).
   - **Max CSN —** It is the most recent CSN the consumer has replayed that was originated from the supplier (identified in the Table Header).
   - **Time Lag —** It shows the time difference between the supplier and the consumer's max CSNs for the changes originated from the supplier (identified in the Table Header). A consumer is in sync with its supplier when its time lag is `0`.
   - **Last Modify Time —** It is roughly the time when the consumer's max CSN was replayed.
   - **Supplier —** This column lists all the suppliers of the consumer.
   - **Sent/Skipped —** Each supplier lists roughly how many changes originated from the supplier (identified in the Table Header) have been replayed or skipped by the consumer. The numbers are kept in suppliers' memory only. They will be cleared if the supplier is restarted.

- **Update Status** — The number is the status code, and the string is the implication of the status code. Watch this column for possible deadlock if all the suppliers complain that they cannot acquire busy replica. It is normal if one of the suppliers is doing an update while the others can't acquire busy replica.

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 makes it simple to search for entries that contain this attribute.

For example, you could use the following `ldapsearch` command:

```
ldapsearch -D adminDN -w password -b "dc=example,dc=com"
"nsds5ReplConflict=*"
```

For performance reasons, if you find that you have many conflicting entries every day, you may want to index the `nsds5ReplConflict` attribute. For information on indexing, refer to chapter 10, "Managing Indexes."

This section contains the procedures for the following conflict resolution procedures:

- Solving Naming Conflicts
- Solving Orphan Entry Conflicts
- Solving Potential Interoperability Problems

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. Each procedure is described below.

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:

```
prompt> ldapmodify -D adminDN -w password

>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:

```
prompt> ldapmodify -D adminDN -w password

>dn: uid=NewValue,dc=example,dc=com
>changetype: modify
>delete: uid
>uid: adamss
>-
>delete: nsds5ReplConflict
>-
```

NOTE You cannot delete the unique identifier attribute `nsuniqueid`.

For more information on the `ldapmodify` command, refer to “Managing Entries from the Command-Line,” on page 57, and *Red Hat Directory Server Configuration, Command, and File Reference*.

The Console does not support editing of multi-valued RDNs. For example, assume you configured two servers in a multi-master mode, created an entry on each server with the same user ID, and changed the new entries' RDN to the `nsuniqueid uid` value. If you attempt to modify this entry from the Console, you get the following error: `Changes cannot be saved for entries with multi-valued RDNs.`

When you open the entry in the advanced mode, you will be able to see that the naming attribute has been set to `nsuniqueid uid`. However, you cannot change or correct the entry by changing the user ID and RDN values to something different. For example, if `jdoe` was the user ID and if you want to change it to `jdoe1`, you cannot do so from the Console. Instead, use the `ldapmodify` command:

```
ldapmodify:
ldapmodify
changetype: modify
replace: uid
uid: jdoe

ldapmodify
changetype: modrdn
newrdn: uid=jdoe1
deleteoldrdn: 1
```

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:

```
prompt> ldapmodify -D adminDN -w password

>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:

```
prompt> ldapmodify -D adminDN -w password

>dn: cn=TempValue,dc=example,dc=com
>changetype: modify
>delete: dc
>dc: pubs
>-
>delete: nsds5ReplConflict
>-
```

NOTE You cannot delete the unique identifier attribute `nsuniqueid`.

3. Rename the entry with the intended attribute-value pair. For example:

```
prompt> ldapmodify -D adminDN -w password

>dn: cn=TempValue,dc=example,dc=com
>changetype: modrdn
>newrdn: dc=NewValue
>deleteoldrdn: 1
```

   By setting the value of the `deleteoldrdn` attribute to `1`, you delete the temporary attribute-value pair `cn=`*TempValue*. If you want to keep this attribute, you can set the value of the `deleteoldrdn` attribute to `0`.

For more information on the `ldapmodify` command, refer to “Managing Entries from the Command-Line,” on page 57, and *Red Hat Directory Server Configuration, Command, and File Reference*.

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 you can modify the glue entry to remove the `glue` object class and the `nsds5ReplConflict` attribute to keep the entry as a normal entry or you can delete the glue entry and its child entries.

- The server creates a minimalistic entry with the `glue` and `extensibleObject` object classes.

  In such cases, you must modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

Solving Potential Interoperability Problems

For reasons of interoperability with applications that rely on attribute uniqueness, such as a mail server, you might need to restrict access to the entries which contain the `nsds5ReplConflict` attribute. If you do not restrict access to these entries, then the applications requiring one attribute only will pick up both the original entry and the conflict resolution entry containing the `nsds5ReplConflict`, and operations will fail.

To restrict access, you need to modify the default ACI that grants anonymous read access, using the following command:

```
ldapmodify -h hostname -D "cn=Directory Manager" -w password

> 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.

For more information on the `ldapmodify` command, refer to “Managing Entries from the Command-Line,” on page 57, and *Red Hat Directory Server Configuration, Command, and File Reference*.

Troubleshooting Replication-Related Problems

This section covers the following:

- Interpreting Error Messages and Symptoms
- Useful Tools

Interpreting Error Messages and Symptoms

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. For details on error log level, check the *Red Hat Directory Server Configuration, Command, and File Reference*.

To change the error log level to `8192`, run the following `ldapmodify` command:

```
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`.

Error Message:

```
agmt=%s (%s:%d) Replica has a different generation ID than the
local data
```

Reason: The consumer specified at the beginning of this message has not been (successfully) initialized yet, or it was initialized from a different root supplier.

Impact: The local supplier will not replicate any data to the consumer.

Remedy: 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.

Error Message:

```
Warning: data for replica %s was reloaded, and it no longer matches
the data in the changelog. Recreating the changelog file. This
could affect replication with replica's consumers, in which case
the consumers should be reinitialized.
```

Reason: 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.

Impact: 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.

Remedy: 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't locate some CSN for a consumer, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

Error Message:

```
agmt=%s(%s:%d): Can't locate CSN %s in the changelog (DB rc=%d). The
consumer may need to be reinitialized.
```

Reason: Most likely the changelog was recreated because of the disk is full or the server ungracefully shutdown.

Impact: 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.

Remedy: 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 Message:

```
Too much time skew
```

Reason: The system clocks on the host machines are extremely out of sync.

Impact: 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.

Remedy: Synchronize the system clocks on the Directory Server host machines. If applicable, run the network time protocol (`ntp`) daemon on those hosts.

Error Message:

```
agmt=%s(%s:%d): Warning: Unable to send endReplication extended
operation (%s)
```

Reason: The consumer is not responding.

Impact: 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.

Remedy: 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.

Symptom:
Changelog is getting too big.

Reason: 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.

Remedy: By default changelog purge is turned off. To turn it on from the command-line, run `ldapmodify` as follows:

```
dn: cn=changelog5,cn=config
changetype: modify
add: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 1d
```

where `1d` means 1 day. Other valid time units are `s`(econds), `m`(inutes), `h`(ours), and `w`(eeks). 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.

Symptom:
The Replication Monitor is not responding. (For information on Replication Monitor, see "Monitoring Replication Status," on page 366.)

Reason: 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.

Remedy: 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
```

Symptom:
In the Replication Monitor, some consumers show just the header of the table. (For information on Replication Monitor, see "Monitoring Replication Status," on page 366.)

Reason: No change has originated from the corresponding suppliers. In this case, the `MaxCSN:` in the header part should be `"None"`.

Remedy: There is nothing wrong if there is no change originated from a supplier.

Useful Tools

The `template-cl-dump.pl` script, which is explained in detail in the *Red Hat Directory Server Configuration, Command, and File Reference*, enables you to 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 `purgeRUV` and `maxRUV`.
- Grep and interpret change sequence numbers (CSNs) in the changelog.
- Get the base-64 encoded changelog from the Directory Server, and then decode the changelog.

Chapter 9

Extending the Directory Schema

Red Hat Directory Server (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 of your requirements, you may need to extend your schema by creating new object classes and attributes.

This chapter describes how to extend your schema in the following sections:

Overview of Extending Schema

When you add new attributes to your schema, you must create a new object class to contain them. Although it may seem convenient to just add the attributes you need to an existing object class that already contains most of the attributes you require, doing so compromises interoperability with LDAP clients.

Interoperability of Directory Server with existing LDAP clients relies on the standard LDAP schema. If you change the standard schema, you will also have difficulties when upgrading your server. For the same reasons, you cannot delete standard schema elements.

For more information on object classes, attributes, and the directory schema, as well as guidelines for extending your schema, refer to *Red Hat Directory Server Deployment Guide*. For information on standard attributes and object classes, see the *Red Hat Directory Server Schema Reference*.

To extend the directory schema, you should proceed in the following order:

1. Create new attributes. See "Creating Attributes," on page 383, for information.
2. Create an object class to contain the new attributes, and add the attributes to the object class. See "Creating Object Classes," on page 387, for information.

Managing Attributes

Through Directory Server Console, you can view all attributes in your schema, and you can create, edit, and delete your attribute extensions to the schema. The following sections describe how to manage attributes:

- Viewing Attributes
- Creating Attributes
- Editing Attributes
- Deleting Attributes

For information on managing object classes, see "Managing Object Classes," on page 385.

Viewing Attributes

To view information about all attributes that currently exist in your directory schema:

1. In the Directory Server Console, select the Configuration tab.
2. In the left navigation tree, select the Schema folder, and then select the Attributes tab in the right pane.

   This tab contains information about all the standard (read-only) and user-defined attributes in the schema.

For information on the fields and lists in the Attributes tab, refer to Table 9-1.

Table 9-1 Attributes Tab Reference

Field or Pane	**Description**
`Name`	The name of the attribute.

Table 9-1 Attributes Tab Reference *(Continued)*

Field or Pane	Description
`OID`	The object identifier of the attribute. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If you do not specify an OID, the Directory Server automatically uses *attribute_name*`-oid`. For example, if you create the attribute `birthdate` without supplying an OID, the Directory Server automatically uses `birthdate-oid` as the OID. For more information about OIDs or to request a prefix for your enterprise, send mail to the IANA (Internet Assigned Number Authority) at iana@iana.org, or visit the IANA website at: http://www.iana.org/.
`Syntax`	The attribute syntax: • Case Ignore String — Indicates that values for this attribute are not case-sensitive. • Case Exact String — Indicates that values for this attribute are case-sensitive. • Distinguished Name — Indicates that values for this attribute are DNs. • Binary — Indicates that values for this attribute are binary. • Telephone Number — Indicates that values for this attribute are in telephone number format. • Integer — Indicates that valid values for this attribute are numbers. Operational attributes are not returned as a result of an `ldapsearch` operation unless they are explicitly specified in the search. Generally, operational attributes are reserved for use by the Directory Server.
`Multi`	If the attribute is multi-valued, an X appears in this column; otherwise, this field is blank. The Directory Server allows more than one instance of a multi-valued attribute per entry.

Creating Attributes

You can use Directory Server Console to create new attributes. After adding new attributes to your schema, you must create a new object class to contain them. See “Creating Object Classes,” on page 387, for information.

To create a new attribute:

1. Display the Attributes tab.

   This procedure is explained in “Viewing Attributes,” on page 382.

2. Click Create.

   The Create Attribute dialog box is displayed.

3. Enter a unique name for the attribute in the Attribute Name text box.

4. Enter an object identifier for the attribute in the Attribute OID (Optional) text box.

   OIDs are described in Table 9-1, on page 382.

5. Select a syntax that describes the data to be held by the attribute from the Syntax drop-down menu.

   Available syntaxes are described in Table 9-1, on page 382.

6. If you want the attribute to be multi-valued, select the Multi-Valued checkbox.

   The Directory Server allows more than one instance of a multi-valued attribute per entry.

7. Click OK.

Editing Attributes

You can only edit attributes you have created. You cannot edit standard attributes.

To edit an attribute:

1. Display the Attributes tab.

   This procedure is explained in "Viewing Attributes," on page 382.

2. Select the attribute that you want to edit in the User Defined Attributes table, and click Edit.

   The Edit Attribute dialog box is displayed.

3. To change the attribute's name, enter a new one in the Attribute Name text box.

4. To change the attribute's object identifier, enter a new one in the Attribute OID (Optional) text box.

   OIDs are described in Table 9-1, on page 382.

5. To change the syntax that describes the data to be held by the attribute, choose a new one from the Syntax drop-down menu.
6. Available syntaxes are described in Table 9-1, on page 382.
7. To make the attribute multi-valued, select the Multi-Valued checkbox.

   The Directory Server allows more than one instance of a multi-valued attribute per entry.
8. When you have finished editing the attribute, click OK.

Deleting Attributes

You can only delete attributes that you have created. You cannot delete standard attributes.

To delete an attribute:

1. Display the Attributes tab.

   This procedure is explained in "Viewing Attributes," on page 382.
2. In the User Defined Attributes table, select the attribute, and click Delete.
3. If prompted, confirm the delete.

   The server immediately deletes the attribute. There is no undo.

Managing Object Classes

You can use Directory Server Console to manage your schema's object classes. Through the Console, you can view all of your schema's object classes and create, edit, and delete your object class extensions to the schema. The following sections describe how to manage object classes:

- Viewing Object Classes
- Creating Object Classes
- Editing Object Classes
- Deleting Object Classes

For information on managing attributes, see "Managing Attributes," on page 382.

Viewing Object Classes

To view information about all object classes that currently exist in your directory schema:

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. In the Object Classes list, select the object class that you want to view.

   This tab displays information about the standard or user-defined object class you selected.

For information on the fields and lists in the Object Classes tab, refer to Table 9-2.

Table 9-2 Object Classes Tab Reference

Field or Pane	**Description**
`Parent`	The parent identifies the object class from which this object class inherits its attributes and structure. For example, the parent object for the `inetOrgPerson` object class is the `organizationalPerson` object. That means that an entry with the object class `inetOrgPerson` must also include the object class `organizationalPerson`. Typically, if you want to add new attributes for user entries, the parent would be the `inetOrgPerson` object class. If you want to add new attributes for corporate entries, the parent is usually `organization` or `organizationalUnit`. If you want to add new attributes for group entries, the parent is usually `groupOfNames` or `groupOfUniqueNames`.
`OID`	The object identifier of the object class. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If you do not specify an OID, the Directory Server automatically uses *ObjectClass_name*`-oid`. For example, if you create the object class `division` without supplying an OID, the Directory Server automatically uses `division-oid` as the OID. For more information about OIDs or to request a prefix for your enterprise, send mail to the IANA (Internet Assigned Number Authority) at iana@iana.org, or visit the IANA website at: `http://www.iana.org/`.
`Object Classes`	This list contains all of the standard and user-defined object classes in the Directory Server schema.
`Required Attributes`	Contains a list of attributes that must be present in entries that use this object class. Includes inherited attributes.

Table 9-2 Object Classes Tab Reference *(Continued)*

Field or Pane	Description
`Allowed Attributes`	Contains a list of attributes that may be present in entries that use this object class. Includes inherited attributes.

Creating Object Classes

You create an object class by giving it a unique name, selecting a parent object for the new object class, and adding required and optional attributes.

To create an object class:

1. Display the Object Classes tab.

   This procedure is explained in "Viewing Object Classes," on page 386.

2. Click Create on the Object Classes tab.

   The Create Object Class dialog box is displayed.

3. Enter a unique name for the object class in the Name text box.

4. Enter an object identifier for the new object class in the OID (Optional) text box.

   OIDs are described in Table 9-2, on page 386.

5. Select a parent object for the object class from the Parent drop-down menu.

   You can choose from any existing object class. See Table 9-2, on page 386, for more information on parent object classes.

6. To add an attribute that *must* be present in entries that use the new object class, highlight the attribute in the Available Attributes list, and then click the Add button to the left of the Required Attributes box.

   You can use either the standard attributes or create new ones. For information, see "Managing Attributes," on page 382.

7. To add an attribute that may be present in entries that use the new object class, highlight the attribute in the Available Attributes list, and then click the Add button to the left of the Allowed Attributes box.

8. To remove an attribute that you previously added, highlight the attribute in the Required Attributes list or the Allowed Attributes list, and then click the corresponding Remove button.

   You cannot remove either allowed or required attributes that are inherited from the parent object classes.

9. When you are satisfied with your object class definition, click OK to dismiss the dialog box.

Editing Object Classes

You can use Directory Server Console to edit object classes that you previously created. You cannot edit a standard object class.

To edit an object class:

1. Display the Object Classes tab.

   This procedure is explained in "Viewing Object Classes," on page 386.

2. Select the object class that you want to edit from the Object Classes list, and click Edit.

   The Edit Object Class dialog box is displayed.

3. To change the name of the object class, enter the new name in the Name text box.

4. To change the object identifier for the object class, enter the new OID in the OID (Optional) text box.

   OIDs are described in Table 9-2, on page 386.

5. To change the parent object for the object class, select the new parent from the Parent pull-down menu.

6. To add an attribute that must be present in entries that use the new object class, highlight the attribute in the Available Attributes list, and then click the Add button to the left of the Required Attributes box.

   You can either use the standard attributes or create new ones. For information, see "Managing Attributes," on page 382.

7. To add an attribute that may be present in entries that use the new object class, highlight the attribute in the Available Attributes list, and then click the Add button to the left of the Allowed Attributes box.

8. To remove an attribute that you previously added, highlight the attribute in the Required Attributes list or the Allowed Attributes list, and then click the corresponding Remove button.

   You cannot remove either allowed or required inherited attributes.

9. When you are satisfied with you the object class definition, click OK to dismiss the dialog box.

Deleting Object Classes

You can delete only object classes that you have created. You cannot delete standard object classes.

To delete an object class:

1. Display the Object Classes tab.

   This procedure is explained in “Viewing Object Classes,” on page 386.

2. Select the object class that you want to remove, and click Delete.
3. If prompted, confirm the delete.

   The server immediately deletes the object class. There is no undo.

Turning Schema Checking On and Off

When schema checking is on, the Directory Server ensures that:

- The object classes and attributes you are 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 you should always run the Directory Server with schema checking turned on. The only case where you might want 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.

To turn schema checking on and off:

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” checkbox; clear it to turn off schema checking.
4. Click Save.

You can also turn schema checking on and off by using the `nsslapd-schemacheck` attribute. For information, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Chapter 10

Managing Indexes

The *Red Hat Directory Server Deployment Guide* introduced the concept of indexing, the costs and benefits, and different types of index shipped with Red Hat Directory Server (Directory Server). This chapter begins with a description of the searching algorithm itself, so as to place the indexing mechanism in context, and then describes how to create, delete, and manage indexes. This chapter contains the following sections:

- About Indexes (page 391)
- Creating Indexes (page 400)
- Deleting Indexes (page 410)
- Managing Indexes (page 418)
- Attribute Name Quick Reference Table (page 422)

About Indexes

This section provides an overview of indexing in Directory Server. It contains the following topics:

- About Index Types
- About Default, System, and Standard Indexes
- Overview of the Searching Algorithm
- Balancing the Benefits of Indexing

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.db3` file.

Directory Server supports the following types of index:

- Presence index (`pres`) — The presence index contains a list of the entries that contain a particular attribute. This index is useful if, for example, you want to examine any entries that contain access control information. Generating an `aci.db3` file that includes a presence index lets you efficiently perform 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`) — The equality index allows you to search efficiently 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`) — The approximate index allows 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`) — The substring index 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
  ```

  would match the common names containing strings such as:

  ```
  Bill Anderson
  Jill Henderson
  Steve Sanderson
  ```

  Similarly, the search for:

  ```
  telephonenumber= *555*
  ```

would return all the entries in your directory with telephone numbers that contain `555`.

- International index — The 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 you apply a matching rule by associating a locale (OID) with the attributes to be indexed.

  For a listing of supported locales and their associated OIDs, refer to Appendix D, "Internationalization." If you want to configure the Directory Server to accept additional matching rules, contact Red Hat Professional Services.

- Browsing (virtual list view) index — The browsing index, or virtual list view 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 branchpoint in the directory tree to improve display performance. You do this through the Directory Server Console or by using the `vlvindex` command-line tool, which is explained in the *Red Hat Directory Server Configuration, Command, and File Reference*.

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.

Overview of Default Indexes

The default indexes can be modified depending on your indexing needs, although you should ensure that no server plug-ins or other servers in your enterprise depend on this index before you remove it.

Table 10-1 lists the default indexes installed with the directory.

Table 10-1 Default Indexes

Attribute	Eq	Pres	Sub	Purpose
`cn`	X	X	X	Improves the performance of the most common types of user directory searches.
`givenName`	X	X	X	Improves the performance of the most common types of user directory searches.

Table 10-1 Default Indexes *(Continued)*

Attribute	Eq	Pres	Sub	Purpose
`mail`	X	X	X	Improves the performance of the most common types of user directory searches.
`mailHost`	X	-	-	Used by a messaging server.
`member`	X	-	-	Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See "Maintaining Referential Integrity," on page 75, for more information.
`owner`	X	-	-	Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See "Maintaining Referential Integrity," on page 75, for more information.
`seeAlso`	X	-	-	Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See "Maintaining Referential Integrity," on page 75, for more information.
`sn`	X	X	X	Improves the performance of the most common types of user directory searches.
`telephoneNumber`	X	X	X	Improves the performance of the most common types of user directory searches.
`uid`	X	-	-	Improves Directory Server performance.
`uniquemember`	X	-	-	Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See "Maintaining Referential Integrity," on page 75, for more information.

Overview of System Indexes

System indexes are indexes that cannot be deleted or modified. They are required by the directory to function properly. Table 10-2 lists the system indexes included with the directory.

Table 10-2 System Indexes

Attribute	Eq	Pres	Purpose
`aci`	-	X	Allows the Directory Server to quickly obtain the access control information maintained in the database.

Table 10-2 System Indexes *(Continued)*

Attribute	Eq	Pres	Purpose
`dnComp`	X	-	Used to help accelerate subtree searches in the directory.
`objectClass`	X	-	Used to help accelerate subtree searches in the directory.
`entryDN`	X	-	Speeds up entry retrieval based on DN searches.
`parentID`	X	-	Enhances directory performance during one-level searches.
`numSubordinates`	-	X	Used by the Directory Server Console to enhance display performance on the Directory tab.
`nsUniqueID`	X	-	Used to search for specific entries.

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 following standard indexes exist by default, and you do not need to generate them:

- `id2entry.db4` — Contains the actual directory database entries. All other database files can be recreated from this one.
- `id2children.db4` — Restricts the scope of *one-level* searches, searches that examine an entry's immediate children.
- `dn.db4` — Controls the scope of *subtree* searches; searches that examine an entry and all the entries in the subtree beneath it.
- `dn2id.db4` — Begins all searches efficiently by mapping an entry's distinguished name to its ID number.

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 Server processes a search request as follows:

1. An LDAP client application, such as the Directory Server Gateway, 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.

3. If the search request for each database attribute can be satisfied by a single index, then the server reads that index to generate a list of potential matches.

    If there is no index for the attribute, the directory generates a candidate list that includes all entries in the database, which makes the search considerably slower.

    If a search request contains multiple attributes, the directory consults multiple indexes and then combines the resulting lists of candidate entries.

4. If there is an index for the attribute, the directory takes the candidate matches from the index files in the form of a series of entry ID numbers.

5. The directory uses the returned entry ID numbers to read the corresponding entries from the `id2entry.db3` file. The Directory Server then examines each of the candidate entries to see if any match the search criteria. The directory returns matching entries to the client as each is found.

    The directory continues until either it has examined all candidate entries or it reaches the limit set in one of the following attributes:

    - `nsslapd-sizelimit` which specifies the maximum number of entries to return from a search operation. If this limit is reached, the directory returns any entries it has located that match the search request, as well as an exceeded size limit error.
    - `nsslapd-timelimit` which specifies the maximum number of seconds allocated for a search request. If this limit is reached, the directory returns any entries it has located that match the search request, as well as an exceeded time limit error.
    - `nsslapd-lookthroughlimit` which specifies the maximum number of entries that the directory will check when examining candidate entries in response to a search request.

- `nsslapd-idlistscanlimit` which specifies the maximum number of entries in an ID list before the list is considered to equal the entire database.

See *Red Hat Directory Server Configuration, Command, and File Reference* for further information about these attributes.

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.

For example:

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.
	Sarette, Alice (SRT ALS)	No match. The codes are not specified in the correct order.

Balancing the Benefits of Indexing

Before you create new indexes, balance the benefits of maintaining indexes against the costs. Keep in mind that:

- 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 will require.

The following example illustrates exactly how time-consuming indexes can become. Consider the procedure for creating a specific attribute:

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, suppose the Directory Server is asked to add 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.
```

Further suppose that the Directory Server is maintaining the following indexes:

- Equality, approximate, and substring indexes for common name and surname attributes.
- Equality and substring indexes for the telephone number attribute.
- Substring indexes for the description attribute.

Then to add this entry to the directory, the Directory Server must perform these steps:

1. Create the common name equality index entry for `John` and `John Doe`.
2. Create the appropriate common name approximate index entries for `John` and `John Doe`.
3. Create the appropriate common name substring index entries for `John` and `John Doe`.
4. Create the surname equality index entry for `Doe`.
5. Create the appropriate surname approximate index entry for `Doe`.
6. Create the appropriate surname substring index entries for `Doe`.
7. Create the telephonenumber equality index entry for `408 555 8834`.
8. Create the appropriate telephonenumber 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.

The example shows that indexing can be costly.

Creating 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.

NOTE Given that this version of Directory Server can operate in either a single or multi-database environment, you need to remember to create your new indexes in every database instance since newly created indexes are not automatically created in the other databases.

However, the same is not entirely true for default indexes because they are automatically present and maintained in subsequent database instances but not added to existing ones. In other words, the directory uses your most recently created set of default indexes in subsequent databases. 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.

As the procedure for creating browsing indexes is different, it is covered in a separate section.

This section contains the following procedures:

- Creating Indexes from the Server Console
- Creating Indexes from the Command-Line
- Creating VLV Indexes from the Server Console
- Creating VLV Indexes from the Command-Line

Creating Indexes from the Server Console

Using the Directory Server Console, you can create presence, equality, approximate, substring, and international indexes for specific attributes.

To create indexes:

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

2. Expand the Data node, expand the suffix of the database you want to index, and select the database.

3. Select the Indexes tab in the right pane.

NOTE	Do not click on the Database Settings node because this will take you to the Default Index Settings window and not the window for configuring indexes per database.

4. If the attribute you want to index is listed in the Additional Indexes table, skip to Step 6. Otherwise, click Add Attribute.

   A dialog box appears containing a list of all of the available attributes in the server schema.

5. Select the attribute you want to index, and click OK.

   The server adds the attribute to the Additional Indexes table.

6. Select the checkbox for each type of index you want to maintain for each attribute.

7. If you want to create an index for a language other than English, enter the OID of the collation order you want to use in the Matching Rules field.

   You can index the attribute using multiple languages by listing 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 Indexes dialog box appears, displaying the status of the index creation and informing you when the indexes have been created. You can click on the Status Logs box to view the status of the indexes created. Once the indexing is complete, click Close to close the Indexes dialog box.

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.

Creating Indexes from the Command-Line

You can create presence, equality, approximate, substring, and international indexes for specific attributes from the command-line.

Creating indexes from the command-line involves two steps:

- Using the `ldapmodify` command-line utility to add a new index entry or edit an existing index entry.
- Running the `db2index.pl` Perl script to generate the new set of indexes to be maintained by the server.

NOTE You cannot create new system indexes because system indexes are hard-coded in Directory Server.

The following sections describe the steps involved in creating indexes.

Adding an Index Entry

Use `ldapmodify` to add the new index attributes to your directory. If you want 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 allows you to centralize configuration information.

For information on the LDIF update statements required to add entries, see “LDIF Update Statements,” on page 65.

For example, assume you want to create presence, equality, and substring indexes for the `sn` (surname) attribute in the `Example1` database.

First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Run the `ldapmodify` command-line utility as follows:

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w
password
```

The `ldapmodify` utility binds to the server and prepares it to add an entry to the configuration file.

Next, you add the following entry for the new indexes:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm
database,cn=plugins,cn=config
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 you want 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 specifies the OID of the Bulgarian collation order.

Specifying an index entry with no value in the `nsIndexType` attribute results in all indexes (except international) being maintained for the specified attribute. For example, suppose instead that you specify the following entry for your new `sn` indexes:

```
dn: cn=sn,cn=index,cn=database_name,cn=ldbm
database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:
```

This new entry results in all indexes for the `sn` (surname) attribute.

You can use the keyword `none` in the `nsIndexType` attribute to specify that no indexes are to be maintained for the attribute. For example, suppose you want to temporarily disable the `sn` indexes you just created on the `Example1` database. You change the `nsIndexType` to `none` as follows:

```
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 collation orders and their OIDs, see Appendix D, "Internationalization."

For more information about the index configuration attributes, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

For more information about the `ldapmodify` command-line utility, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

NOTE You should 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 `userid` attribute. See Table 10-3, on page 422, for a list of all primary and alias attribute names.

Running the db2index.pl Script

Once you have created an indexing entry or added 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. Once you run the script, the new set of indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `db2index.pl` Perl script:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Run the `db2index.pl` Perl script.

   For more information about using this Perl script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example generates an index using the `db2index.pl` UNIX shell script:

```
db2index.pl -D "cn=Directory Manager" -w passsword  -n
ExampleServer -t sn
```

The following table describes the `db2index.pl` options used in the examples:

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 into which you are importing the data.
`-t`	Specifies the name of the attribute contained by the index.

For more information about the `db2index.pl` Perl script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Creating VLV Indexes from the Server Console

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).

To create a VLV index from the Console:

1. In the Directory Server Console, select the Directory tab.
2. In the left navigation tree, create a suffix where you want to create the VLV index. For instance, for views based on the locality (`l`) attribute, name this organizational unit `Location Views`.
3. Right-click on `ou=Location Views`, and select New>Other.
4. Select `nsview` from the New Object menu, and hit okay.
5. In the Property Editor window, hit the “Add Value” button, and add the organization unit object class.
6. Name the organization unit according to how you want to organize the VLVs you will create. For instance, `ou=Sunnyvale`. Make the `ou` attribute the naming attribute.

7. Hit the "Add Attribute" button, and add the `nsviewfilter` attribute.
8. Create a filter that reflects the views you will create. For example:

   ```
   (l=Sunnyvale)
   ```

9. Hit okay to close the attributes box, and hit okay again to save the new VLV index.

The new index is immediately active for any new data that you add to your 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 "Adding a Browsing Index Entry," on page 407, and "Setting Access Control for VLV Information," on page 410.

A browsing index is a type of VLV index that organizes the entries listed into alphabetical order, making it easier to find entries. To create a browsing index using the Directory Server Console:

1. In the Directory Server Console, select the Directory tab.
2. In the left navigation tree (for example, `People`), select the entry for which you want 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. You can click on the Status Logs box to view the status of the indexes created.

4. Click Close to close the Create Browsing Index dialog box.

   The new index is immediately active for any new data that you add to your directory. You do not have to restart your server.

Creating VLV Indexes from the Command-Line

Creating a browsing index or virtual list view (VLV) index from the command-line involves these steps:

- Using `ldapmodify` to add new browsing index entries or edit existing browsing index entries.
- Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server.

- Ensuring that access control on VLV index information is set appropriately.

The following sections describe the steps involved in creating browsing indexes.

Adding a Browsing Index Entry

The type of browsing index entry you want to create depends on the type of `ldapsearch` attribute sorting you want to accelerate. It is important to take the following into account:

- The scope of the search (base, one, sub).

  For more information on the `ldapsearch -s` option, which allows you to specify the scope of searches, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

- The base of the search (the entry you want to use as a starting point for the search).

  For more information on the `ldapsearch -b` option, which allows you to specify the base of searches, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

- The attributes you want to sort.
- The filter of the search.

  For more information on specifying filters for searches, see Appendix B, "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, you want to create a browsing index to accelerate an `ldapsearch` on the entry `ou=People,dc=example,dc=com` 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 `one`, and
- the sorting order for the returned attributes is `cn`, `givenname`, `o`, `ou`, and `sn`.

First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Run the `ldapmodify` command-line utility as follows:

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w
password
```

The `ldapmodify` utility binds to the server and prepares it to add an entry to the configuration file.

Next, you need to add two browsing index entries which define your browsing index.

The first entry you add specifies the base, scope, and filter of the browsing index:

```
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))
```

The `cn` contains the browsing index identifier, which specifies the entry on which you want to create the browsing index; in this example, the `ou=People,dc=example,dc=com` entry. We recommend you use the `dn` of the entry for your 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 base for the search you want to accelerate is `1`. A search base 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))`.

The second entry you add specifies the sorting order you want 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.

Running the vlvindex Script

Once you have created 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 you run the script, the new set of browsing indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `vlvindex` script:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Stop the server:

   ```
   ./stop-slapd
   ```

3. Run the `vlvindex` script.

   For more information about using this script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

4. Restart the server.

   ```
   ./start-slapd
   ```

This example generates a browsing index using the `vlvindex` UNIX shell script:

```
vlvindex -n Example1 -T "by MCC ou=people dc=example dc=com"
```

The following table describes the `vlvindex` options used in the examples:

Option	Description
`-n`	Name of the database containing the entries to index.

Option	Description
`-T`	Browsing index identifier to use to create browsing indexes.

For more information about the `vlvindex` script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Setting Access Control for VLV Information

The default access control for the VLV index information is to allow anyone who has authenticated. If a site requires anonymous users to use the VLV index information, modify the access control set for `cn: VLV Request Control` in the Directory Server's configuration.

1. Change to the configuration directory: *serverRoot*`/slapd-`*serverID*`/config`
2. In a text editor, open the `dse.ldif` file.
3. Locate `oid=2.16.840.1.113730.3.4.9`; you should see these lines:

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectClass: top
objectClass: directoryServerFeature
oid: 2.16.840.1.113730.3.4.9
cn: VLV Request Control
aci: (targetattr != "aci")(version 3.0; acl "VLV Request
Control"; allow( read, search, compare, proxy ) userdn =
"ldap:///all" ;)
creatorsName: cn=server,cn=plugins,cn=config
modifiersName: cn=server,cn=plugins,cn=config
...
```

4. Change `"ldap://all"` to `"ldap://anyone"` and save your changes.

Deleting Indexes

This section describes how to delete presence, equality, approximate, substring, international, and browsing indexes for specific attributes.

NOTE Because this version of Directory Server can operate in either a single or multi-database environment, you have to delete any unwanted indexes from every database instance. Any default indexes you delete will not be deleted from previous sets of indexes on existing database instances.

As the procedure for deleting browsing indexes is different, it is covered in a separate section. This section contains the following procedures:

- Deleting Indexes from the Server Console
- Deleting Indexes from the Command-Line
- Deleting Browsing and VLV Indexes from the Server Console
- Deleting Browsing and VLV Indexes from the Command-Line

CAUTION You must not delete system indexes because deleting them can significantly affect Directory Server performance. System indexes are located in the `cn=index,cn=instance,cn=ldbm database,cn=plugins,cn=config` entry and the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

Take care when deleting default indexes since this can also affect how Directory Server works.

For further information on system and default indexes, see the *Red Hat Directory Server Deployment Guide*.

Deleting Indexes from the Server Console

Using the Directory Server Console you can delete indexes you have created, indexes used by other server applications such as a messaging or web server, and default indexes. You cannot delete system indexes.

To delete indexes using the Directory Server Console:

1. In the Directory Server Console, select the Configuration tab.
2. Expand the Data node and expand the suffix associated with the database containing the index. Select the database from which you want to delete the index.

3. Locate the attribute containing the index you want to delete. Clear the checkbox under the index.

   If you want to delete all indexes maintained for a particular attribute, select the attribute's cell under Attribute Name, and click Delete Attribute.

4. Click Save.

   A Delete Index warning dialog box appears asking you to confirm that you want to delete the index.

5. Click Yes to delete the index.

6. The Delete Browsing Index dialog box appears displaying the status of the index deletion. You can click on the Status Logs button to view the status of the indexes deleted.

7. Once the indexing is complete, click on Close to close the Delete Browsing Index box.

Deleting Indexes from the Command-Line

You can browsing indexes, or virtual list view (VLV) indexes, using the `ldapdelete` command-line utility as follows:

- Delete an entire index entry or delete unwanted index types from an existing index entry using the `ldapdelete` command-line utility.
- Generate the new set of indexes to be maintained by the server using the `db2index.pl` script.

The following sections describe the steps involved in deleting an index.

Deleting an Index Entry

Use the `ldapdelete` command-line utility to delete either the entire indexing entry or the unwanted index types from an existing entry.

If you want to delete the indexes for a particular database, you remove your index entry 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.

To delete a default index, remove it from the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

For example, you want to delete presence, equality, and substring indexes for the `sn` attribute on the database named `Example1`.

You want to delete the following entry:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm
database,cn=plugins,cn=config
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
```

To run the `ldapdelete` command-line utility, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Perform the `ldapdelete` as follows:

```
ldapdelete -D "cn=Directory Manager" -w password -h ExampleServer
-p845 "cn=sn,cn=index,cn=Example1,dn=ldbm database,
cn=plugins,dn=config"
```

The following table describes the `ldapdelete` options used in the example:

Option	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` option.
`-h`	Specifies the name of the host on which the server is running.
`-p`	Specifies the port number that the server uses.

For full information on `ldapdelete` options, refer to the *Red Hat Directory Server Configuration, Command, and File Reference.*

Once you have deleted this entry, the presence, equality, and substring indexes for the `sn` attribute will no longer be maintained by the `Example1` database.

Running the db2index.pl Script

Once you have deleted an indexing entry or deleted some of the index types from an indexing entry, run the `db2index.pl` script to generate the new set of indexes to be maintained by the Directory Server. Once you run the script, the new set of indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `db2index.pl` Perl script:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Run the `db2index.pl` Perl script.

   For more information about using the `db2index.pl` Perl script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This example generates a new set of indexes to be maintained by the server, using the `db2index.pl` UNIX shell script:

```
db2index.pl -D "cn=Directory Manager" -w password -n Example1
```

The following table describes the `db2index.pl` options used in the examples:

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 into which you are importing the data.

For more information about the `db2index.pl` Perl script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Deleting Browsing and VLV Indexes from the Server Console

Using the Directory Server Console, you can delete browsing indexes and VLV indexes.

To delete a virtual list:

1. In the Directory Server Console, select the Directory tab.
2. Select the virtual list that you want to delete, such as `ou=Sunnyvale,ou=LocationViews,dc=example,dc=com` (to delete all the virtual lists, you can delete the entire subsuffix, `ou=LocationViews,dc=example,dc=com`).
3. Right-click on the entry, and select Delete from the drop-down menu. You can also select the entry, and select Object>Delete from the tool menu at the top.
4. A dialog box appears asking you to confirm that you want to delete the entry. Click Yes to delete.

To delete a browsing index:

1. In the Directory Server Console, select the Directory tab.
2. Select the entry from which you want to delete the index in the navigation tree, and select Delete Browsing Index from the Object menu. You can also select and right-click the entry of the index you want 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 that you want to delete the index. Click Yes to delete.
4. The Delete Browsing Index dialog box appears displaying the status of the index deletion.

Deleting Browsing and VLV Indexes from the Command-Line

Deleting a browsing index or virtual list view (VLV) index from the command-line involves two steps:

- Using the `ldapdelete` to delete browsing index entries or edit existing browsing index entries.
- Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server.

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.

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, you remove your 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, you want to delete a browsing index for accelerating `ldapsearch` operations on the entry `ou=People,dc=example,dc=com` 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, you need to delete the two corresponding browsing index entries which follow:

```
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))
```

and

```
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
```

To run the `ldapdelete` command-line utility, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Perform the `ldapdelete` as follows:

```
ldapdelete -D "cn=Directory Manager" -w password -h ExampleServer
-p 845 "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"
```

The following table describes the `ldapdelete` options used in the example:

Option	**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` option.
`-h`	Specifies the name of the host on which the server is running.
`-p`	Specifies the port number that the server uses.

For full information on `ldapdelete` options, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

Once you have deleted these two browsing index entries, the browsing index for accelerating `ldapsearch` operations on the entry `ou=People dc=example dc=com` 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` will no longer be maintained by the `Example1` database.

Running the vlvindex Script

Once you have deleted browsing indexing entries or deleted 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. Once you run the script, the new set of browsing indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `vlvindex` script:

1. From the command-line, change to the following directory:

   serverRoot`/slapd-`*serverID*`/`

2. Stop the server:

   ```
   ./stop-slapd
   ```

3. Run the `vlvindex` script.

   For more information about using the `vlvindex` script, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

4. Restart the server:

   ```
   ./start-slapd
   ```

This example recreates the indexes using the `vlvindex` UNIX shell script:

```
vlvindex -n Example1 -T "by MCC ou=people dc=example dc=com"
```

The following table describes the `vlvindex` options used in the examples.

Option	Description
`-n`	Name of the database containing the entries to index.
`-T`	Browsing index identifier to use to create browsing indexes.

For more information about the `vlvindex` script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Managing Indexes

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 (see "About Indexes," on page 391, for details).

The redesigned secondary index structure greatly improves write, search, and indexing operations. The following sections examine enhanced indexing operations and the BerkeleyDB design, searching and the old All IDs Threshold, and migrating and compatibility with previous versions of Directory Server.

Indexing Performance

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 backend and opaque to the storage manager.
- The storage manager structures (Btrees), which were opaque to the Directory Server backend 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 8Kbytes. 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 BerkeleyDB feature 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. As a result:

- 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 (8Kbytes) to twice the size of one ID (4bytes).
- For short ID lists, storage efficiency, and in most cases performance, is improved because only the storage manager metadata 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.

Search Performance

For each entry ID list, there is a size limit that is globally applied to all index keys managed by the server. In previous versions of Directory Server, this limit was called the All IDs Threshold. Because maintaining large ID lists in memory can affect performance, the All IDs Threshold set a limit on how large a single entry ID list could get. When a list hit a certain pre-determined size, the search would treat it as if the index contained the entire directory.

The difficulty in setting the All IDs Threshold hurt peformance. If the threshold was too low, too many searches examined every entry in the directory. If it was too high, too many large ID lists had to be maintained in memory.

The problems addressed by the All IDs Threshold are no longer present because of the efficiency of entry insertion, modification, and deletion in the BerkeleyDB design. The All IDs Threshold is removed for database write operations, and every ID list is now maintained accurately.

Since loading a long ID list from the database can significantly reduce search performance, the configuration parameter, `nsslapd-idlistscanlimit`, sets a limit on the number of IDs that are read before a key is considered to match the entire primary index. `nsslapd-idlistscanlimit` is analagous to the All IDs Threshold, but it only applies to the behavior of the server's search code, not the content of the database.

idlistscanlimit

When the server uses indexes in the processing of a search operation, it is possible that one index key matches a large number of entries. For example, consider a search for `objectclass=inetorgperson` in a directory that contained one million `inetorgperson` entries. When the server reads the `inetorgperson` index key in the `objectclass` index, it would find one million matching entries. In cases like this, it is usually more efficient simply to conclude in the index lookup phase of the search operation processing that all the entries in the database match the query. This causes subsequent search processing to scan the entire database content, checking each entry as to whether it matches the search filter. The idea is that the time required to fetch the index keys is not worthwhile; the search operation is likely to be processed more efficiently by omitting the index lookup.

Directory Server implements this search optimization as follows: when examining an index, if more than a certain number of entries are found, the server will stop reading the index and mark the search as unindexed with respect to that particular index.

The threshold number of entries is called the `idlistscanlimit` and is configured with the `nsslapd-idlistscanlimit` configuration attribute. The default value is `4000`, which is designed to give good performance for a common range of database sizes and access patterns. Typically, it is not necessary to change this value. However, in rare circumstances it may be possible to improve search performance with a different value. For example, lowering the value will improve performance for searches that will otherwise eventually hit the default limit of 4000. This might, of course, reduce performance for other searches that would benefit from indexing. Conversely, increasing the limit could improve performance for searches that were previously hitting the limit. With a higher limit, these searches could benefit from indexing where previously they did not.

For more information on search limits for the server, see “Overview of the Searching Algorithm,” on page 395.

Backwards Compatibility and Migration

While current versions of Directory Server can support the old database design, only the new design is supported for this and later releases of Directory Server.

Upon startup, the server will read the database version from the DBVERSION file, which contains the text `Netscape-ldbm/6.2` (old database version) and `Netscape-ldbm/7.1` (new database format). If the file indicates that the old format is used, then the old code is selected for the database. Because the DBVERSION file stores everything per-backend, it is theoretically possible to have different database formats for different individual backends. However, the old database format is not recommended.

All databases must be migrated to Directory Server 7.1 when the system is upgraded. Migration is supported for Directory Server 6.x versions. Version 7.1 can be installed “over” the previous releases, resulting in a working server that supports the new database design. Migrating the databases is a lengthy operation; do not do this at the time of installation.

For releases earlier than version 6.11, it is recommended that the databases be dumped, and Directory Server installed fresh.

For more information on migrating databases, see chapter 6, “Migrating from Previous Versions,” in the *Red Hat Directory Server Installation Guide.*

Also, the index sizes can be larger than in previous releases, so you may want to increase your database cache size. To reconfigure your cache size, refer to "nsslap-dbcachesize" in the *Red Hat Directory Server Configuration, Command, and File Reference.*

Attribute Name Quick Reference Table

Table 10-3 lists all attributes which have a primary or real name as well as an alias. When creating indexes be sure to use the primary name.

Table 10-3 Attribute Name Quick Reference Table

Attribute Primary Name	Attribute Alias
`dn`	`distinguishedName`
`cn`	`commonName`
`sn`	`surName`
`c`	`countryName`
`l`	`localityName`
`st`	`stateOrProvinceName`
`street`	`streetAddress`
`o`	`organization`
`ou`	`organizationalUnitName`
`facsimileTelephoneNumber`	`fax`
`uid`	`userId`
`mail`	`rfc822mailbox`
`mobile`	`mobileTelephoneNumber`
`pager`	`pagerTelephoneNumber`
`co`	`friendlyCountryName`
`labeledUri`	`labeledUri`
`ttl`	`timeToLive`
`dc`	`domainComponent`
`authorCn`	`documentAuthorCommonName`
`authorSn`	`documentAuthorSurname`

Table 10-3 Attribute Name Quick Reference Table *(Continued)*

`drink`	`favoriteDrink`

Chapter 11

Managing SSL and SASL

To provide secure communications over the network, Red Hat Directory Server (Directory Server) includes the LDAPS communications protocol. LDAPS is the standard LDAP protocol, but it runs on top of Secure Sockets Layer (SSL). Directory Server also allows “spontaneous” secure connections over otherwise-insecure LDAP ports, using Start TLS (Transport Layer Security).

Directory Server also supports SASL authentication using the GSS-API mechanism, allowing Kerberos, rather than certificates, to authenticate sessions and encrypt data.

This chapter describes how to use SSL and SASL with your Directory Server in the following sections:

Introduction to SSL in the Directory Server

The Directory Server supports SSL/TLS 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. You can use SSL/TLS with simple authentication (bind DN and password) or with certificate-based authentication.

Using SSL with simple authentication ensures confidentiality and data integrity. The benefits of using a certificate to authenticate to the Directory Server instead of a bind DN and password include:

- Improved efficiency — When you are using applications that prompt you once for your 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. This is because certificate-based authentication uses public-key cryptography. As a result, bind credentials cannot be intercepted across the network.

The Directory Server is capable of simultaneous SSL and non-SSL communications. This means that you do not have to choose between SSL or non-SSL communications for your Directory Server; you can use both at the same time. You can also utilize the Start TLS extended operation to allow SSL/TLS secure communication over a regular (insecure) LDAP port.

Directory Server also supports SASL client authentication; see "Introduction to SASL," on page 445, for more information.

Enabling SSL: Summary of Steps

To configure your Directory Server to use LDAPS, follow these steps:

1. Obtain and install a certificate for your Directory Server, and configure the Directory Server to trust the certification authority's (CA's) certificate.

   For information, see""Obtaining and Installing Server Certificates," on page 428.

2. Turn on SSL in your directory.

   For information, see "Starting the Server with SSL Enabled," on page 434.

3. Configure the Administration Server to connect to an SSL-enabled Directory Server.

   For information, see *Managing Servers with Red Hat Console*.

4. Optionally, ensure that each user of the Directory Server obtains and installs a personal certificate for all clients that will authenticate with SSL.

   For information, see "Configuring LDAP Clients to Use SSL," on page 443.

For a complete description of SSL, Internet security, and certificates, check the appendixes included in *Managing Servers with Red Hat Console*.

Command-Line Functions for Start TLS

You can specify that LDAP operations such as `ldapmodify`, `ldapsearch`, and `ldapdelete` use SSL/TLS when communicating with an SSL-enabled server or to use certificate authentication. Using the command-line options, you can also specify or enforce Start TLS, which which allows a secure connection to be enabled on a cleartext port after a session has been initiated.

In the following example, a network administrator enforces Start TLS for a search for Mike Connor's identification number:

```
ldapsearch -p 389 -ZZZ -P certificateDB -N certificate_name -s base -b
"uid=mconnors" "(attribute=govIdNumber)"
```

where `-ZZZ` enforces Start TLS, *certificateDB* gives the filename and path to the certificate database, and *certificate_name* is the certificate.

NOTE The `-ZZZ` command enforces the use of Start TLS, and the server must respond that a Start TLS command was successful. If you use the `-ZZZ` command and the server does not support Start TLS, the operation is aborted immediately.

For information on the command-line options available, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Troubleshooting Start TLS

With the `-ZZ` option, the following errors could occur:

- If there is no certificate database, the operation fails. See "Obtaining and Installing Server Certificates," on page 428, for information on using certificates.
- If the server does not support Start TLS, the connection proceeds in cleartext. To enforce the use of Start TLS, use the `-ZZZ` command option.
- If the certificate database does not have the Certifying Authority (CA) certificate, the connection proceeds in cleartext. See "Obtaining and Installing Server Certificates," on page 428, for information on using certificates.

With the `-ZZZ` option, the following errors could occur, causing the Start TLS operation to fail:

- If there is no certificate database. See "Obtaining and Installing Server Certificates," on page 428, for information on using certificates.
- If the certificate database does not have the Certifying Authority (CA) certificate. See "Obtaining and Installing Server Certificates," on page 428, 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"`.

Obtaining and Installing Server Certificates

This section describes the process of creating a certificate database, obtaining and installing a certificate for use with your Directory Server, and configuring Directory Server to trust the certification authority's (CA) certificate.

This process is a necessary first step before you can turn on SSL in your directory. If you have already completed these tasks, see "Starting the Server with SSL Enabled," on page 434.

Obtaining and installing certificates consists of the following steps:

- Step 1: Generate a Certificate Request
- Step 2: Send the Certificate Request to the Certificate Authority
- Step 3: Install the Certificate
- Step 4: Trust the Certificate Authority

- Step 5: Confirm That Your New Certificates Are Installed

You will use the Certificate Request Wizard to generate a certificate request (Step 1) and send it to a Certificate Authority (Step 2). You then use the Certificate Install Wizard to install the certificate (Step 3) and to trust the Certificate Authority's certificate (Step 4).

These wizards automate the process of creating a certificate database and of installing the key-pair.

Step 1: Generate a Certificate Request

To generate a certificate request and send it to a CA:

1. In the Directory Server Console, select the Tasks tab, and click Manage Certificates.

   The Manage Certificates window is displayed.

2. Select the Server Certs tab, and click the Request button.

   The Certificate Request Wizard is displayed.

3. Click Next.

4. Enter the Requestor Information in the blank text fields, then click Next.

   Enter the following information:

   - **Server Name** — Enter the fully qualified hostname of the Directory Server as it is used in DNS lookups; for example, `dir.example.com`.
   - **Organization** — Enter the legal name of your company or institution. Most CAs require you to verify this information with legal documents such as a copy of a business license.
   - **Organizational Unit** — *Optional*. Enter a descriptive name for your organization within your company.
   - **Locality** — *Optional*. Enter your company's city name.
   - **State or Province** — Enter the full name of your company's state or province (no abbreviations).
   - **Country** — Select the two-character abbreviation for your country's name (ISO format). The country code for the United States is US.

5. Enter the password that will be used to protect the private key, and click Next.

   The Next field is greyed out until you supply a password. When you click Next, the Request Submission dialog box is displayed.

6. Select Copy to Clipboard or Save to File to save the certificate request information that you must send to the Certificate Authority.

7. Click Done to dismiss the Certificate Request Wizard.

Once you have generated the request, you are ready to send it to the CA.

Step 2: Send the Certificate Request

Follow these steps to send the certificate information to the CA:

1. Use your email program to create a new email message.

2. Copy the certificate request information from the clipboard or the saved file into the body of the message.

   The content will look similar to the following example:

   ```
   -----BEGIN NEW CERTIFICATE REQUEST-----
   MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCVXMxEzARBgNVBAgTCkNBTElGT1J
   OSUExLDAqBgVBAoTI25ldHNjYXBlIGNvbW11bmljYXRpb25zIGNvcnBvcmF
   0aW9uMRwwGgYDVQQDExNtZWxsb24ubmV0c2NhcGUuY29tMIGfMA0GCSqGSI
   b3DQEBAQUAA4GNADCBiQKBgQCwAbskGh6SKYOgHy+UCSLnm3ok3X3u83Us7
   ug0EfgSLR0f+K41eNqqRftGR83emqPLDOf0ZLTLjVGJaH4Jn4l1gG+JDf/n
   /zMyahxtV7+mT8GOFFigFfuxaxMjr2j7IvELlxQ4IfZgWwqCm4qQecv3G+N
   9YdbjveMVXW0v4XwIDAQABoAAwDQYK
   -----END NEW CERTIFICATE REQUEST-----
   ```

3. Send the email message to the CA.

Once you have emailed your request, you must wait for the CA to respond with your certificate. Response time for requests varies. For example, if your CA is internal to your company, it may only take a day or two to respond to your request. If your selected CA is external to your company, it could take several weeks to respond to your request.

When the CA sends a response, be sure to save the information in a text file. You will need the data when you install the certificate.

You should also back up the certificate data in a safe location. If your system ever loses the certificate data, you can reinstall the certificate using your backup file.

Once you receive your certificate, you are ready to install it in your server's certificate database.

Step 3: Install the Certificate

To install a server certificate:

1. In the Directory Server Console, select the Tasks tab, and click Manage Certificates.

   The Manage Certificates window is displayed.

2. Select the Server Certs tab, and click Install.

   The Certificate Install Wizard is displayed.

3. Choose one of the following options for the certificate location, 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 text file you created, and paste it in this field. For example:

```
-----BEGIN CERTIFICATE-----
MIICMjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMx
IzAhBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0wGwYDVQQLExRX
aWRnZXQgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVz
dCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3WhcNOTgwMzI2MDIzMzU3WjBP
MQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTmV0c2NhcGUgRGlyZWN0b3J5IFB1Ymxp
Y2F0aW9uczEWMBQGA1UEAxMNZHVgh49dq2itLmNvbTBaMA0GCSqGSIb3
-----END CERTIFICATE-----
```

4. Check that the certificate information displayed is correct, and click Next.

5. Specify a name for the certificate, and click Next.

6. Verify the certificate by providing the password that protects the private key.

   This password is the same as the one you provided in "Step 1: Generate a Certificate Request," on page 431.

Now that you have installed your certificate, you need to configure your server to trust the Certificate Authority from which you obtained the server's certificate.

Step 4: Trust the Certificate Authority

Configuring your Directory Server to trust the certificate authority consists of obtaining your CA's certificate and installing it into your server's certificate database. This process differs depending on the certificate authority you use. Some commercial CAs provide a web site that allows you to automatically download the certificate. Others will email it to you upon request.

Once you have the CA certificate, you can use the Certificate Install Wizard to configure the Directory Server to trust the Certificate Authority.

1. In the Directory Server Console, select the Tasks tab, and click Manage Certificates.

   The Manage Certificates window is displayed.

2. Go to the CA Certs tab, and click Install.

   The Certificate Install Wizard is displayed.

3. If you saved the CA's certificate to a file, enter the path in the field provided. If you received the CA's certificate via email, copy and paste the certificate, including the headers, into the text field provided. Click Next.
4. Check that the certificate information that is displayed is correct, and click Next.
5. Specify a name for the certificate, and click Next.
6. Select the purpose of trusting this Certificate Authority (you can select both):
   - **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.
7. Click Done to dismiss the wizard.

Once you have installed your certificate and trusted the CA's certificate, you are ready to activate SSL. However, you should first make sure that the certificates have been installed correctly.

Step 5: Confirm That Your New Certificates Are Installed

1. In the Directory Server Console, select the Tasks tab, and click Manage Certificates.

The Manage Certificates window is displayed.

2. Select the Server Certs tab.

A list of all the installed certificates for the server is displayed.

3. Scroll through the list. You should find the certificates you installed.

Your server is now ready for SSL activation.

NOTE When you renew a certificate using the Certificate Wizard, the text on the introduction screen (step 1) doesn't clearly indicate that the process is renewal and not requesting a new certificate. Also, the requestor information (step 2) doesn't get filled automatically.

Using certutil

The Directory Server has a command-line tool, `certutil`, which allows you to create self-signed CA and client certificates, certificate databases, and keys. `certutil` can also be downloaded from `ftp://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/`.

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 by default:

```
cd serverRoot/alias
```

2. Create a password file for your security token password:

```
vi pwdfile.txt
secretpwd
```

3. Create a “noise” file for your encryption mechanism:

```
vi noise.txt
dsadasdasdasdadasdasdasdasdsadfwerwerjfdksdjfksdlfhjsdk
```

4. Create the `key3.db` and `cert8.db` databases:

   ```
   /serverRoot/shared/bin/certutil -N -d . -f pwdfile.txt
   ```

5. Generate the encryption key:

   ```
   /serverRoot/shared/bin/certutil -G -d . -z noise.txt -f
   pwdfile.txt
   ```

6. Generate the self-signed certificate:

   ```
   /serverRoot/shared/bin/certutil -S -n "CA certificate" -s
   "cn=CAcert" -x -t "CT,," -m 1000 -v 120 -d . -z noise.txt -f
   pwdfile.txt
   ```

7. Generate the server certificate:

   ```
   /serverRoot/shared/bin/certutil -S -n "Server-Cert" -s
   "cn=server-cert" -c "CA certificate" -t "u,u,u" -m 1001 -v
   120 -d . -z noise.txt -f pwdfile.txt
   ```

   To generate a client certificate to use with applications other than the Directory Server, run the same command as for the Directory Server certificate.

8. Copy the `key3.db` and `cert8.db` you created to the default databases created at Directory Server installation:

   ```
   mv key3.db slapd-server-key3.db
   mv cert8.db slapd-server-cert8.db
   ln -s slapd-server-key3.db key3.db
   ln -s slapd-server-cert8.db cert8.db
   ```

9. Run `pk12util` to convert the certificate database to `pkcs12` format, so it is accessbile by the Directory Server:

   ```
   /serverRoot/shared/bin/pk12util -d . -o cert.pk12 -n
   Server-Cert
   ```

The certificates created by `certutil` are automatically available in the Encryption tab of the Console; there is not need to import them.

Starting the Server with SSL Enabled

Most of the time, you want your server to run with SSL enabled. If you temporarily disable SSL, make sure you re-enable it before processing transactions that require confidentiality, authentication, or data integrity.

Before you can activate SSL, you must create a certificate database, obtain and install a server certificate, and trust the CA's certificate, as described in "Obtaining and Installing Server Certificates," on page 428.

NOTE On SSL-enabled servers, be sure to check the file permissions on certificate-database files, key-databases 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.

Enabling SSL Only in the Directory Server:

1. Obtain and install CA and server certificates.
2. Set the secure port you want the server to use for SSL communications.

   The encrypted port number that you specify must not be the same port number you use for normal LDAP communications. By default, the standard port number is `389`, and the secure port is `636`.

   a. Change the secure port number in the Configuration>Settings tab of the Directory Server Console. Save.

   b. Restart the Directory Server. It will restart still with the regular port.
3. In the Directory Server Console, select the Configuration tab, and then select the topmost 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" checkbox.
5. Check the "Use this Cipher Family" checkbox.
6. Select the certificate that you want to use from the drop-down menu.
7. Click Cipher Settings.

   The Cipher Preference dialog box is displayed. By default, all ciphers are selected.
8. Set your preferences for client authentication.

   - **Do not allow client authentication —** With this option, the server will ignore 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 "Using Certificate-Based Authentication," on page 441.
- **Require client authentication** — With this option, the server requests authentication from the client.

If you are only enabling SSL in the Directory Server, do not select "Require client authentication" checkbox.

NOTE If you are using certificate-based authentication with replication, then you must configure the consumer server either to allow or to require client authentication.

---

9. You can further configure the server to verify the authenticity of requests by selecting the "Check hostname against name in certificate for outbound SSL connections" option. The server does this verification by matching the hostname against the value assigned to the common name (`cn`) attribute of the subject name in the certificate being presented for authentication.

   By default, this feature is disabled. If it's enabled and if the hostname 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 hostname doesn't 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 (Can't contact LDAP
   server)
   ```

   It is recommended that you enable 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. You must restart from the command-line.

Enabling SSL in the Directory Server, Admin Server, and Console

1. Obtain server certificates and CA certs, and install them on the Directory Server.

2. Obtain and install server and CA certificates on the Administration Server.

   It is important that the Administration Server and Directory Server have a CA certificate in common so that they can trust the other's certificates.

3. If you have not installed the servers as `root`, it is necessary to change the secure port setting from the default `636` to a number above `1024`.

   a. Change the secure port number in the Configuration>Settings tab of the Directory Server Console. Save.

   b. Restart the Directory Server. It will restart still with the regular port.

4. In the Configuration tab of the Directory Server Console, highlight the server name at the top of the table, and select the Encryption tab.

5. Select the "Enable SSL" checkbox.

6. Check the "Use this Cipher Family" checkbox.

7. Select the certificate that you want to use from the drop-down menu.

8. Click Cipher Settings.

   The Cipher Preference dialog box is displayed. By default, all ciphers are selected.

9. Set your preferences for client authentication.

   ❍ **Do not allow client authentication** — With this option, the server will ignore 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 "Using Certificate-Based Authentication," on page 441.

   ❍ **Require client authentication** — With this option, the server requests authentication from the client.

NOTE If you are using certificate-based authentication with replication, then you must configure the consumer server either to allow or to require client authentication.

10. You can further configure the server to verify the authenticity of requests by selecting the “Check hostname against name in certificate for outbound SSL connections” option. The server does this verification by matching the hostname against the value assigned to the common name (`cn`) attribute of the subject name in the certificate being presented for authentication.

    By default, this feature is disabled. If it’s enabled and if the hostname 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 hostname doesn’t 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 (Can’t contact LDAP
    server)
    ```

    It is recommended that you enable this option to protect Directory Server’s outbound SSL connections against a Man in the Middle (MITM) attack.

11. Check the "Use SSL in the Console” box. Hit “Save.”
12. In the Administration Server Console, select the Configuration tab. Select the Encryption tab, check the “Enable SSL” checkbox, and fill in the appropriate certificate information.
13. In the Configuration DS tab, change the port number to the new Directory Server secure port information. See “Changing Directory Server Port Numbers,” on page 39, for more information. Do this even if you are using the default port of `636`. Check the “Secure Connection” checkbox.
14. In the User DS tab, select the “Set User Directory” radio button, and fill in the new Directory Server secure port information, the LDAP URL, and the user database information. Check the “Secure Connection” checkbox.
15. Save the new SSL settings, Configuration DS, and User DS information in the Administration Server.

16. Restart the Admin Server. You must start the server from the command-line.

17. Restart the Directory Server. You must start the server from the command-line.

When you restart the Console, be certain that the address reads `https`; otherwise, the operation will time out, unable to find the Admin Server since it is running on a secure connection. When you successfully connect, a dialog box will appear, asking you to accept the certificate. Click OK to accept the certificate (you may choose whether to accept it only for that session or for always).

Creating a Password File

You can create a password file to store your certificate password. By placing your certificate database password in a file, you can start your server from the server console and also allow your server to restart automatically when running unattended.

CAUTION This password is stored in cleartext within the password file, so its usage represents a significant security risk. Do not use a password file if your server is running in an unsecured environment.

The password file must be placed in the following location:

serverRoot`/alias/slapd-`*serverID*`-pin.txt`

where *serverID* is the identifier you specified for the server when you installed it.

You need to include the token name and password in the file, as follows:

```
Token:mypassword
```

For example:

```
Internal (Software) Token:mypassword
```

Setting Security Preferences

You can choose the type of ciphers you want to use for SSL communications. 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. For a more complete discussion of algorithms and their strength, see *Managing Servers with Red Hat Console*.

When a client initiates an 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. Your server needs to be able to use the ciphers that will be used by client applications connecting to the server.

Directory Server provides the following SSL 3.0 ciphers:

- RC4 cipher with 40-bit encryption and MD5 message authentication.
- RC2 cipher with 40-bit encryption and MD5 message authentication.
- No encryption, only MD5 message authentication.
- DES with 56-bit encryption and SHA message authentication.
- RC4 cipher with 128-bit encryption and MD5 message authentication.
- Triple DES with 168-bit encryption and SHA message authentication.
- FIPS DES with 56-bit encryption and SHA message authentication. This cipher meets the FIPS 140-1 U.S. government standard for implementations of cryptographic modules.
- FIPS Triple DES with 168-bit encryption and SHA message authentication. This cipher meets the FIPS 140-1 US government standard for implementations of cryptographic modules.

To select the ciphers you want the server to use:

1. Make sure SSL is enabled for your server.

   For information, see “Starting the Server with SSL Enabled,” on page 434.

2. In the Directory Server Console, 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.

   This displays the current server encryption settings.

4. Click Cipher Settings.

   The Cipher Preference dialog box is displayed.

5. In the Cipher Preference dialog box, specify which ciphers you want your server to use by selecting them from the list, and click OK.

   Unless you have a security reason not to use a specific cipher, you should select all of the ciphers, except for `none,MD5`.

6. In the Encryption tab, click Save.

CAUTION Avoid selecting the `none,MD5` cipher because the server will use this option if no other ciphers are available on the client. It is not secure because encryption doesn't occur.

In order to continue using the Red Hat Console with SSL, you must select at least one of the following ciphers:

- RC4 cipher with 40-bit encryption and MD5 message authentication.
- No encryption, only MD5 message authentication.
- DES with 56-bit encryption and SHA message authentication.
- RC4 cipher with 128-bit encryption and MD5 message authentication.
- Triple DES with 168-bit encryption and SHA message authentication.

Using Certificate-Based Authentication

Directory Server allows you to use certificate-based authentication for the command-line tools (which are LDAP clients) and for replication communications. Certificate-based authentication can occur between:

- An LDAP client connecting to the Directory Server.
- A Directory Server connecting to another Directory Server (replication or chaining).

NOTE When specifying the key and certificate database filenames, you may use absolute or relative paths. If using relative paths, ensure that they are relative to the server root (for example, `alias/slapd-phonebook-cert8.db` and `alias/slapd-phonebook-key3.db`).

The name of the certificate database has been changed from `cert7.db` to `cert8.db`. Directory Server automatically converts the `cert7.db` to `cert8.db` and uses the new file. However, the `dse.ldif` file may not show the new database name. For example, you may still see this entry:

```
nsCertfile: alias/slapd-testDir-cert7.db
```

If you want the database filename change reflected in the `dse.ldif` file, manually edit the filename in the `dse.ldif` file.

Setting up Certificate-Based Authentication

To set up certificate-based authentication, you must:

1. Create a certificate database for the client and the server or for both servers involved in replication.

   In the Directory Server, the certificate database creation automatically takes place when you install a certificate. For information on creating a certificate database for a client, see "Configuring LDAP Clients to Use SSL," on page 443.

2. Obtain and install a certificate on both the client and the server or on both servers involved in replication.

3. Enable SSL on the server or on both servers involved in replication.

   For information on enabling SSL, refer to "Starting the Server with SSL Enabled," on page 434.

NOTE If Red Hat Console connects to Directory Server over SSL, selecting "Require client authentication" disables communication. This is because, although Red Hat Console supports SSL, it does not have a certificate to use for client authentication.

4. Map the certificate's distinguished name to a distinguished name known by your directory.

   This allows you to set access control for the client when it binds using this certificate. This mapping process is described in *Managing Servers with Red Hat Console*.

Allowing/Requiring Client Authentication

If you have configured Red Hat Console to connect to your Directory Server using SSL *and* your Directory Server *requires* client authentication, you can no longer use Red Hat Console to manage server applications. You will have to use the appropriate command-line utilities instead.

However, if at a later date you wish to change your directory configuration to no longer *require* but *allow* client authentication, so that you can use Red Hat Console, you must follow these steps:

1. Stop Directory Server.

   For information on stopping and starting the server from the command-line, see "Starting and Stopping the Server from the Command-Line," on page 38.

2. Modify the `cn=encryption,cn=config` entry by changing the value of the `nsSSLClientAuth` attribute from `required` to `allowed`.

   For information on modifying entries from the command-line, see chapter 2, "Creating Directory Entries."

3. Start Directory Server.

   You can now start Red Hat Console.

Configuring LDAP Clients to Use SSL

If you want all the users of your Directory Server to use SSL or certificate-based authentication when they connect using LDAP client applications, you must make sure they perform the following tasks:

- Create a certificate database.
- Trust the Certificate Authority (CA) that issues the server certificate.

These operations are sufficient if you want to ensure that LDAP clients recognize the server's certificate. However, if you also want LDAP clients to use their own certificate to authenticate to the directory, make sure that all your directory users obtain and install a personal certificate.

NOTE Some client applications do not verify that the server has a trusted certificate.

1. On the client system, obtain a client certificate from the CA.
2. On your client system, install your client certificate.

   Regardless of how you receive your certificate (either in email or on a web page), there should be a link that you click to install the certificate.

   Make sure you record the certificate information that is sent to you in a file. In particular, you must know the subject DN of the certificate because you must configure the server to map it to an entry in the directory. Your client certificate will be similar to:

   ```
   -----BEGIN CERTIFICATE-----
   MIICMjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBh
   MCVVMxIzAhBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0w
   GwYDVQQLExRXaWRnZXQgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdC
   BUZXN0IFRlc3QgVGVzdCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3
   WhcNOTgwMzI2MDIzMzU3WjBPMQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTm
   V0c2NhcGUgRGlyZWN0b3
   -----END CERTIFICATE-----
   ```

3. You must convert the client certificate into its binary format using the `certutil` utility. To do this:

   `certutil -L -d` *certdbPath* `-n` *userCertName* `-r >` *userCert.bin*

   where *certdbPath* is the location of your certificate database, *userCertName* is the name you gave to your certificate when you installed it, and *userCert.bin* is the name you must specify for the output file that will contain the certificate in the binary format.

4. On the server, map the subject DN of the certificate that you obtained to the appropriate directory entry by editing the `certmap.conf` file.

   This procedure is described in *Managing Servers with Red Hat Console*.

NOTE Do not map your certificate-based-authentication certificate to a distinguished name under `cn=monitor`. If you map your certificate to a DN under `cn=monitor`, your bind will fail. Map your 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 `userCertificate` and `userCertificate;binary` attributes.

5. In the Directory Server, modify the directory entry for the user who owns the client certificate to add the `userCertificate` attribute.

   a. Select the Directory tab, and navigate to the user entry.

   b. Double click the user entry, and use the Property Editor to add the `userCertificate` attribute, with the `binary` subtype.

      When you add this attribute, instead of an editable field, the server provides a Set Value button.

   c. Click Set Value.

      A file selector is displayed. Use it to select the binary file you created in Step 3.

   For information on using the Directory Server Console to edit entries, refer to “Modifying Directory Entries,” on page 51.

You can now use SSL with your LDAP clients. For information on how to use SSL with `ldapmodify`, `ldapdelete`, and `ldapsearch`, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

Introduction to SASL

Directory Server supports LDAP client authentication through the Simple Authentication and Security Layer (SASL), an alternative to SSL/TLS and a native way for some applications to share information securely.

SASL is a framework, meaning it sets up a system that allows different mechanisms authenticate a user to the server, depending on what mechanism is enabled in both client and server applications.

SASL can also set up a security layer for an encrypted session. Directory Server utilizes the GSS-API mechanism to encrypt data during sessions.

NOTE SASL data encryption is not supported for client connections that use SSL/TLS.

Authentication Mechanisms

Directory Server support the following SASL encryption mechanisms:

- EXTERNAL

  The EXTERNAL authentication mechanism is utilized by services such as SSL/TLS. It can be used with public keys for strong authentication.

- 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 plaintext passwords and does protect against plaintext 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 via function calls that can be used to wrap and unwrap payload data. This allows LDAP clients to authenticate with the server using Kerberos version 5 credentials.

NOTE GSS-API and, thus, Kerberos are only supported on platforms that have GSS-API support.

DIGEST-MD5 and GSS-API are “shared secret” mechanisms. This means that the server challenge 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.

SASL Identity Mapping

When processing a SASL bind request, the server matches, or maps, the SASL user ID used to authenticate to the Directory Server with an LDAP entry stored within the server.

If the user ID clearly corresponds to the LDAP entry for a person, it is possible to configure the Directory Server to map the authentication DN automatically to the entry DN. Every branch in the directory tree has a default map, and customized maps can be created. During a bind attempt, a randomly selected custom map is applied. If only one user identity is returned, the bind is successful; if none or more than one are returned, then the next custom map is tried, and so on, until the default is tried. If no map works, then the bind fails.

NOTE SASL proxy authorization is not supported in Directory Server; therefore, the server will ignore any SASL `authzid` supplied by the client.

SASL is 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 contain three attributes, `nsSaslMapRegexString`, `nsSaslMapBaseDNTemplate`, and `nsSaslMapFilterTemplate`. The `nsSaslMapping` object class sets these identity mapping parameters.

The `nsSaslMapRegexString` attribute sets variables of the form `\1`, `\2`, `\3`, etc., for bind IDs which are filled into the template attributes during a search. For example, assume the `nsSaslMapping` is set up as follows:

```
dn: cn=mymap,cn=mapping,cn=sasl,cn=config
objectclass:top
objectclass:nsSaslMapping
cn: mymap
nsSaslMapRegexString: (.*)@(.*)\.(.*)
nsSaslFilterTemplate: (objectclass=inetOrgPerson)
nsSaslBaseDNTemplate: uid=\1,ou=people,dc=\2,dc=\3
```

A bind attempt with `mconnors@example.com` as the regular expression would "fill in" the base DN template with `uid=mconnors,ou=people,dc=example,dc=com` as the authentication ID, and authentication would proceed from there.

You could also write a broader mapping scheme, such as the following:

```
objectclass: top
objectclass: nsSaslMapping
cn: mymap2
nsSaslMapRegexString: .*
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=&)
```

This will match any user ID and map to the result of the the subtree search with base `ou=People,dc=example,dc=com` and filter `cn=`*userId*.

Legacy Identity Mapping

Older versions of Directory Server did support limited SASL mechanisms, EXTERNAL and DIGEST-MD5.

These mechanisms have simple username-based identies, so the server implements a simple identity mapping scheme using the `uid` to find the corresponding directory entries.

A user binds with an authentication DN such as `uid=bjensen,ou=people,dc=example,dc=com`, and the server searches across the entire directory contents, looking for an entry with a corresponding `uid`. This identity mapping is hard-coded and cannot be changed.

Because Kerberos has more complicated identities (see "Realms," on page 450), the new regular expression-based mapping scheme was added. In processing a bind request, the server first tries to apply any regular expression mapping, if configured. If no match is found, then the server tries to apply legacy mapping.

Configuring SASL Identity Mapping from the Console

1. In the Console, open the Directory Server.
2. Open the "Configuration" tab.
3. Select the "SASL Mapping" tab.
4. To add new SASL identities, select the "Add" button, and fill in the required values.
5. Before you can modify a SASL identity, you must have saved that identity. Then, you can click on the "Modify" button, and a text box appears with the current values. Change the values you want, and then close and hit "Save."
6. To delete a SASL identity, highlight it and hit the "Delete" button. When you hit "Save," then as dialog box will appear asking if you want to delete the specified identities. Hit "yes" to continue with the save.
7. Before you hit save, you can undo a modify or delete by selecting the "Reset" button, which will revert to the last saved SASL identities configuration list.

Configuring SASL Identity Mapping from the Command-Line

To configure SASL identity mapping from the command-line, use the `ldapsearch` utility to configure an identity mapping scheme, such as the following:

```
objectclass: top
objectclass: nsSaslMapping
cn: mymap2
nsSaslMapRegexString: .*
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=&)
```

This will match any user ID and map to the result of the the subtree search with base `ou=People,dc=example,dc=com` and filter `cn=`*userId*.

For more information on the `ldapsearch` utility, see "Using ldapsearch," on page 590.

Configuring Kerberos

Kerberos v5 must be deployed on your system to utilize the GSS-API mechanism for SASL authentication. Table 11-1 summarizes the Kerberos applications supported by various platforms. GSS-API must be enabled as a SASL mechanism in the Directory Server to take advantage of Kerberos services.

Table 11-1 Supported Kerberos Systems

Linux	MIT Kerberos version 5
HP-UX 11i	HP Kerberos version 2.1
Sun Solaris	SEAM 1.0.1

Realms

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.

Realms are used by the server to associate the DN of the client in the following form, which looks like an LDAP URL:

`uid=`*user_name*`/[`*server_instance*`],cn=`*realm*`,cn=`*mechanism*`,cn=auth`

NOTE Kerberos systems treat the Kerberos realm as the default realm; other systems default to the server.

Mike Connors in the `engineering` realm of the European division of `example.com` would have the following association if he tried to access a different server, such as `cyclops`:

```
uid=mconnors/cn=Europe.example.com,
cn=engineering,cn=gssapi,cn=auth
```

Babs Jensen in the `accounting` realm of `US.example.com` would not have to specify *server_instance*:

```
uid=bjensen,cn=accounting,cn=gssapi,cn=auth
```

If realms are supported by the mechanism and the default realm was not used, *realm* must be specified; otherwise, it is omitted. Currently, only GSS-API supports the concept of realms.

Configuring the KDC Server

To use GSS-API, the user first obtains a ticket granting ticket (TGT). The ticket and the ticket's lifetime are parameters in the `kdc` server configuration in the `/etc/krb5/krb5.conf` file. See "Example," on page 451.

NOTE The HP server and client are separate packages with their own configuration. The server stores config files in `/opt/krb5`. The client is classic MIT and uses `/etc/krb5.conf`. You need to configure both to have a working Kerberos system.

In order to respond to Kerberos operations, the Directory Server requires access to its own cryptographic key. This key is read by the Kerberos libraries that the server calls, via GSSAPI, and the details of how it is found are implementation-dependent. However, in current releases of the supported Kerberos implementations, the mechanism is the same: the key is read from a file called a *keytab* file. This file is created by the Kerberos administrator by exporting the key from the KDC. Either the system default keytab file (typically `/etc/krb5.keytab`) is used, or a service-specific keytab file determined by the value of the `KRB5_KTNAME` environment variable.

The Directory Server uses the service name `ldap`. Its Kerberos principal is `ldap/`*host-fqdn*`@`*realm*. A key with this identity must be stored in the server's keytab in order for Kerberos to work.

For information on setting up the service key, see your Kerberos documentation.

Example

Code Example 11-1 is an example code for a KDC server configured with the `company.example.com` realm.

Code Example 11-1 Configuring an Example KDC Server

```
[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
```

Chapter 12

Monitoring Server and Database Activity

This chapter describes monitoring database and Red Hat Directory Server (Directory Server) logs. This chapter contains the following sections:

- Viewing and Configuring Log Files (page 453)
- Manual Log File Rotation (page 461)
- Monitoring Server Activity (page 461)
- Monitoring Database Activity (page 468)
- Monitoring Database Link Activity (page 473)

For information on using SNMP to monitor your Directory Server, see chapter 13, "Monitoring Directory Server Using SNMP."

Viewing and Configuring Log Files

Directory Server provides three types of logs to help you better manage your directory and tune performance. These logs include:

- Access Log
- Error Log
- Audit Log

The following aspects are common to the configuration of all types of logs:

- Defining a log file creation policy.
- Defining a log file deletion policy.

The following sections describe how to define your log file creation and deletion policy and how to view and configure each type of log.

NOTE When the server is not running, you cannot read the logs using the Directory Server Console. However, you can read them using the Administration Server Console:

1. From your browser, access: `http://`*hostname*`:`*admin_server_port*
2. At the login prompt, use the admin login ID and password.
3. Click the link for Red Hat Administration Express.

Defining a Log File Rotation Policy

If you want the directory to periodically archive the current log and start a new one, you can define a log file rotation policy from Directory Server Console. You can configure the following parameters:

- The the access mode or file permissions with which log files are to be created. The default value is `600`. The valid values are any combination of `000` to `777`, as they mirror *numbered* or *absolute* UNIX file permissions. That is, the value must be a combination of a 3-digit number, the digits varying from `0` through `7`:

  `0` - None
  `1` - Execute only
  `2` - Write only
  `3` - Write and execute
  `4` - Read only
  `5` - Read and execute
  `6` - Read and write
  `7` - Read, write, and execute

  In the 3-digit number, the first digit represents the owner's permissions, the second digit represents the group's permissions, and the third digit represents everyone's permissions. When changing the default value, keep in mind that `000` will not allow access to the logs and that allowing write permissions to everyone can result in the logs being overwritten or deleted by anyone.

  The newly configured access mode will only affect new logs that are created; the mode will be set when the log rotates to a new file.

- The total number of logs you want the directory to keep. When the directory reaches this number of logs, it deletes the oldest log file in the folder before creating a new log. The default is `10` logs. Do not set this value to `1`. If you do, the directory will not rotate the log, and the log will grow indefinitely.
- The maximum size (in MB) for each log file. If you don't want to set a maximum size, type `-1` in this field. The default is `100` MB. Once a log file reaches this maximum size (or the maximum age defined in the next step), the directory archives the file and starts a new one. If you set the maximum number of logs to `1`, the directory ignores this attribute.
- How often the directory archives the current log file and creates a new one by entering a number of minutes, hours, days, weeks, or months. You can also rotate logs at a particular time of the day; for example, everyday at midnight. The default is every day. If you set the maximum number of logs to `1`, the directory ignores this attribute.

Each log file inludes a title, which identifies the server version, hostname, and port, for ease of archiving or exchanging log files. The title is of the form:

```
Red Hat-Directory/version  build_number
hostname:port  (instance_directory)
```

For example, the first couple of lines of any log files generated by a Directory Server instance may show lines similar to these:

```
Red Hat-Directory/7.1 B2003.188.1157
myhost.example.com:389 (/opt/redhat-ds/servers/slapd-ds71)
```

Defining a Log File Deletion Policy

If you want the directory to automatically delete old archived logs, you can define a log file deletion policy from the Directory Server Console.

NOTE The log deletion policy only makes sense if you have previously defined a log file rotation policy. Log file deletion will not work if you have just one log file. The server evaluates the log file deletion policy at the time of log rotation.

You can configure the following parameters:

- The maximum size of the combined archived logs. When the maximum size is reached, the oldest archived log is automatically deleted. If you don't want to set a maximum size, type `-1` in this field. The default is `500` MB. This parameter is ignored if the number of log files is set to `1`.
- The minimum amount of free disk space. When the free disk space reaches this minimum value, the oldest archived log is automatically deleted. The default is `5` MB. This parameter is ignored in the number of log files is set to `1`.
- The maximum age of log files. When a log file reaches this maximum age, it is automatically deleted. The default is `1` month. This parameter is ignored in the number of log files is set to `1`.

Access Log

The access log contains detailed information about client connections to the directory. This section contains the following procedures:

- Viewing the Access Log
- Configuring the Access Log

Viewing the Access Log

To view the access log:

1. In the Directory Server Console, select the Status tab; then, in the navigation tree, expand the Logs folder, and select the Access Log icon.

   A table displays a list of the last 25 entries in the access log.

2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.

NOTE Continuous log refresh does not work well with log files over 10Mbytes.

3. To view an archived access log, select it from the Select Log pull-down menu.
4. To display a different number of messages, enter the number you want to view in the "Lines to show" text box, and then click Refresh.

5. You can display messages containing a string you specify. To do this, enter the string in the “Show only lines containing” text box, and then click Refresh.

Configuring the Access Log

You can configure a number of settings to customize the access log, including where the directory stores the access log and the creation and deletion policies.

You can also disable access logging for the directory. You may do this because the access log can grow very quickly; every 2,000 accesses to your directory will increase your access log by approximately 1 MB. However, before you turn off access logging, consider that the access log provides beneficial troubleshooting information.

To configure the access log for your directory:

1. In the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder, and select the Access Log icon.

   The access log configuration attributes are displayed in the right pane.

2. To enable access logging, select the Enable Logging checkbox.

   Clear this checkbox if you do not want the directory to maintain an access log. Access logging is enabled by default.

3. Enter the full path and filename you want the directory to use for the access log in the Log File field. The default path is:

   serverRoot`/slapd-`*serverID*`/logs/access`

4. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see “Defining a Log File Rotation Policy,” on page 454.

5. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see “Defining a Log File Deletion Policy,” on page 455.

6. When you have finished making changes, click Save.

Error Log

The error log contains detailed messages of errors and events the directory experiences during normal operations. This section contains the following procedures:

- Viewing the Error Log
- Configuring the Error Log

Viewing the Error Log

To view the error log:

1. In the Directory Server Console, select the Status tab; then, in the navigation tree, expand the Logs folder, and select the Error Log icon.

   A table displays a list of the last 25 entries in the error log.

2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.

NOTE	Continuous log refresh does not work well with log files over 10Mbytes.

3. To view an archived error log, select it from the Select Log pull-down menu.
4. To specify a different number of messages, enter the number you want to view in the “Lines to show” text box, and click Refresh.
5. You can display messages containing a string you specify. To do this, enter the string in the “Show only lines containing” text box, and click Refresh.

Configuring the Error Log

You can change several settings for the error log, including where the directory stores the log and what you want the directory to include in the log.

To configure the error log:

1. In the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder ,and select the Error Log icon.

   The error log configuration attributes are displayed in the right pane.

2. Select the Error Log tab in the right pane.

3. To enable error logging, select the Enable Logging checkbox.

   Clear this checkbox if you do not want the directory to maintain an error log. Error logging is enabled by default.

4. Enter the full path and filename you want the directory to use for the error log in the Log File field. The default path is:

   serverRoot`/slapd-`*serverID*`/logs/errors`

5. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see "Defining a Log File Rotation Policy," on page 454.

6. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see "Defining a Log File Deletion Policy," on page 455.

7. If you want to set the log level, Ctrl+click the options you want the directory to include in the Log Level list box.

   For more information about log level options, see "Log Level" in the *Red Hat Directory Server Configuration, Command, and File Reference*.

   Changing these values from the defaults may cause your error log to grow very rapidly, so it is recommended that you do not change your logging level unless you are asked to do so by Red Hat Technical Support.

8. When you have finished making changes, click Save.

Audit Log

The audit log contains detailed information about changes made to each database as well as to server configuration. This section contains the following procedures:

- Viewing the Audit Log
- Configuring the Audit Log

Viewing the Audit Log

Before you can view the audit log, you must enable audit logging for the directory. See "Configuring the Audit Log," on page 460, for information.

To view the audit log:

1. In the Directory Server Console, select the Status tab. Then, in the navigation tree, expand the Logs folder, and select the Audit Log icon.

   A table displays a list of the last 25 entries in the audit log.

2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.

NOTE	Continuous log refresh does not work well with log files over 10Mbytes.

3. To view an archived audit log, select it from the Select Log pull-down menu.

4. To display a different number of messages, enter the number you want to view in the "Lines to show" text box, and click Refresh.

5. You can display messages containing a string you specify. To do this, enter the string in the "Show only lines containing" text box, and click Refresh.

Configuring the Audit Log

You can use the Directory Server Console to enable and disable audit logging and to specify where the audit log file is stored.

To configure audit logging:

1. In the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder, and select the Audit Log icon.

   The audit log configuration attributes are displayed in the right pane.

2. To enable audit logging, select the Enable Logging checkbox.

   To disable audit logging, clear the checkbox. By default, audit logging is disabled.

3. Enter the full path and filename you want the directory to use for the audit log in the field provided. The default path is:

   serverRoot`/slapd-`*serverID*`/logs/audit`

4. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see "Defining a Log File Rotation Policy," on page 454.

5. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see "Defining a Log File Deletion Policy," on page 455.

6. When you have finished making changes, click Save.

Manual Log File Rotation

The Directory Server supports automatic log file rotation for all three logs. However, you can manually rotate log files if you have not set automatic log file creation or deletion policies. By default, access, error, and audit log files can be found in the following location:

serverRoot`/slapd-`*serverID*`/logs/`

To manually rotate log files:

1. Shut down the server.

   See "Starting and Stopping the Directory Server," on page 37, for instructions.

2. Move or rename the log file you are rotating in case you need the old log file for future reference.

3. Restart the server.

   See "Starting and Stopping the Directory Server," on page 37, for instructions.

Monitoring Server Activity

You can monitor your Directory Server's current activities from either the Directory Server Console or the command-line. You can also monitor the activity of the caches for all of your database. This section contains the following information:

- Monitoring Your Server from the Directory Server Console
- Monitoring Your Server from the Command-Line

Monitoring Your Server from the Directory Server Console

This section contains information about using the Directory Server Console to monitor your server and the information available to you in the performance monitor.

Viewing the Server Performance Monitor

To monitor your server's activities using Directory Server Console:

1. In the Directory Server Console, select the Status tab. In the navigation tree, select Performance Counters.

   The Status tab in the right pane displays current information about server activity. If the server is currently not running, this tab will not provide performance monitoring information.

2. Click Refresh to refresh the current display. If you want the server to continuously update the displayed information, select the Continuous checkbox.

Overview of Server Performance Monitor Information

The server provides monitoring information as described in the following sections:

- General Information (Server)
- Resource Summary
- Current Resource Usage
- Connection Status
- Global Database Cache Information

General Information (Server)

The server provides the following general information:

- Server version — Identifies the current server version.

- Configuration DN — Identifies the distinguished name that you must use as a search base to obtain these results using the `ldapsearch` command-line utility. This field should read `cn=monitor`.
- Data version — Provides identification information for the server's data. Usually the information shown here is only relevant if your server supplies replicas to consumer servers. The data version information is supplied as follows:
  - Server hostname.
  - Server port number.
  - Database generation number. *Obsolete.* A unique identifier that is created only when you create your directory database without a machine data entry in the LDIF file.
  - Current changelog number. This is the number corresponding to the last change made to your directory. This number starts at one and increments by one for each change made to the database.
- Startup time on server — Date and time the server was started.
- Current time on server — Displays the current date and time on the server.

Resource Summary

The Resource Summary table displayed by the Console provides resource-specific information listed in Table 12-1.

Table 12-1 Server Performance Monitoring - Resource Summary

Resource	**Usage since startup**	**Average per minute**
Connections	Total number of connections to this server since server startup.	Average number of connections per minute since server startup.
Operations Initiated	Total number of operations initiated since server startup. Operations include any client requests for server action, such as searches, adds, and modifies. Often, multiple operations are initiated for each connection.	Average number of operations per minute since server startup.
Operations Completed	Total number of operations completed by the server since server startup.	Average number of operations per minute since server startup.
Entries Sent to Clients	Total number of entries sent to clients since server startup. Entries are sent to clients as the result of search requests.	Average number of entries sent to clients per minute since server startup.

Table 12-1 Server Performance Monitoring - Resource Summary *(Continued)*

Resource	Usage since startup	Average per minute
Bytes Sent to Clients	Total number of bytes sent to clients since server startup.	Average number of bytes sent to clients per minute since server startup.

Current Resource Usage

The Resource Summary table in Directory Server Console provides resource-specific information listed in Table 12-2.

Table 12-2 Server Performance Monitoring - Current Resource Usage

Resource	Current total
Active Threads	Current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
Open Connections	Total number of open connections. Each connection can account for multiple operations, and therefore multiple threads.
Remaining Available Connections	Total number of remaining connections that the server can concurrently open. This number is based on the number of currently open connections and the total number of concurrent connections that the server is allowed to open. In most cases, the latter value is determined by the operating system and is expressed as the number of file descriptors available to a task.
Threads Waiting to Write to Client	Total number of threads waiting to write to the client. Threads may not be immediately written when the server must pause while sending data to a client. Reasons for a pause include a slow network, a slow client, or an extremely large amount of information being sent to the client.
Threads Waiting to Read from Client	Total number of threads waiting to read from the client. Threads may not be immediately read if the server starts to receive a request from the client, and then the transmission of that request is halted for some reason. Generally, threads waiting to read are an indication of a slow network or client.
Databases in Use	Total number of databases being serviced by the server.

Connection Status

The Connection Status table in the Directory Server Console provides the following information about the amount of resources in use by each currently open connection:

Table 12-3 Server Performance Monitoring - Connection Status

Table Header	Description
Time opened	Indicates the time on the server when the connection was initially opened.
Started	Indicates the number of operations initiated by this connection.
Completed	Indicates the number of operations completed by the server for this connection.
Bound as	Indicates the distinguished name used by the client to bind to the server. If the client has not authenticated to the server, the server displays `not bound` in this field.
Read/Write	Indicates whether the server is currently blocked for read or write access to the client. Possible values include: • Not blocked. Indicates that the server is idle, actively sending data to the client, or actively reading data from the client. • Blocked. Indicates that the server is trying to send data to the client or read data from the client but cannot. The probable cause is a slow network or client.

Global Database Cache Information

The Global Database Cache Information table in the Directory Server Console contains the following information:

Table 12-4 Server Performance Monitoring - Global Database Cache

Table Header	Description
Hits	Indicates the number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
Tries	The total number of requests performed on your directory since server startup.
Hit Ratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
Pages read in	Indicates the number of pages read from disk into the cache.
Pages written out	Indicates the number of pages written from the cache back to disk.
Read-only page evicts	Indicates the number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.

Table 12-4 Server Performance Monitoring - Global Database Cache *(Continued)*

Table Header	Description
Read-write page evicts	Indicates the number of read-write pages discarded from the cache to make room for new pages. This value differs from Pages Written Out in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.

Monitoring Your Server from the Command-Line

You can monitor your Directory Server's current activities from any LDAP client by performing a search operation with the following characteristics:

- Search for attribute `objectClass=*`
- Search base: `cn=monitor`
- Search scope: `base`

For example:

```
ldapsearch -h directory.example.com -s base -b "cn=monitor"
"(objectclass=*)"
```

For information on searching the Directory Server, see "Using ldapsearch," on page 590.

The monitoring attributes for your server are found in the `cn=monitor,cn=config` entry.

When you monitor your server's activities using `ldapsearch`, you see the following information:

- `version`: Identifies the directory's current version number.
- `threads`: Current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
- `connection:`*fd*`:`*opentime*`:`*opsinitiated*`:`*opscompleted*`:`*binddn*`:[rw]`: Provides the following summary information for each open connection (only available if you bind to the directory as the Directory Manager):
    - `fd` — The file descriptor used for this connection.
    - `opentime` — The time this connection was opened.

    - `opsinitiated` — The number of operations initiated by this connection.
    - `opscompleted` — The number of operations completed.
    - `binddn`—The distinguished name used by this connection to connect to the directory.
    - `rw` — The field shown if the connection is blocked for read or write.

    By default, this information is available to you only if you bind to the directory as the Directory Manager. However, you can change the ACI associated with this information to allow others to access the information.

- `currentconnections`: Identifies the number of connections currently in service by the directory.
- `totalconnections`: Identifies the number of connections handled by the directory since it started.
- `dtablesize`: Shows the number of file descriptors available to the directory. Each connection requires one file descriptor: one for every open index, one for log file management, and one for `ns-slapd` itself. Essentially, this value lets you know about how many more concurrent connections can be serviced by the directory. For more information on file descriptors, refer to your operating system documentation.
- `readwaiters`: Identifies the number of threads waiting to read data from a client.
- `opsinitiated`: Identifies the number of operations the server has initiated since it started.
- `opscompleted`: Identifies the number of operations the server has completed since it started.
- `entriessent`: Identifies the number of entries sent to clients since the server started.
- `bytessent`: Identifies the number of bytes sent to clients since the server started.
- `currentime`: Identifies the time when this snapshot of the server was taken. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
- `starttime`: Identifies the time when the server started. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
- `nbackends`: Identifies the number of back ends (databases) the server services.
- `backendmonitordn`: Identifies the DN of each directory database.

Monitoring Database Activity

You can monitor your database's current activities from Directory Server Console or from the command-line. This section contains the following information:

- Monitoring Database Activity from the Server Console
- Monitoring Databases from the Command-Line

Monitoring Database Activity from the Server Console

This section describes how you can use Directory Server Console to view the database performance monitors and what sort of information the performance monitors provide.

Viewing Database Performance Monitors

To monitor your database's activities:

1. In the Directory Server Console, select the Status tab. In the navigation tree, expand the Performance Counters folder, and select the database that you want to monitor.

   The tab displays current information about database activity. If the server is currently not running, this tab will not provide performance monitoring information.

2. Click Refresh to refresh the currently displayed information. If you want the directory to continuously update the displayed information, select the Continuous checkbox, and then click Refresh.

Overview of Database Performance Monitor Information

The directory provides database monitoring information as described in the following sections:

- General Information (Database)
- Summary Information Table
- Database Cache Information Table

- Database File-Specific Table

General Information (Database)

The directory provides the following general database information:

- Database — Identifies the type of database that you are monitoring.
- Configuration DN — Identifies the distinguished name that you must use as a search base to obtain these results using the `ldapsearch` command-line utility.

Summary Information Table

The Summary Information table provides the following information:

Table 12-5 Database Performance Monitoring - Summary Information

Performance Metric	Current Total
Readonly status	Indicates whether the database is currently in read-only mode. Your database is in read-only mode when the `readonly` attribute is set to `on`.
Entry cache hits	Indicates the total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
Entry cache tries	Indicates the total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against your server since server startup.
Entry cache hit ratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever a search operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, you can increase the number of entries that the directory maintains in the entry cache by increasing the value of the “Maximum Entries in Cache” attribute. See “Tuning Database Performance,” on page 486, for information on changing this value using the Server Console.
Current entry cache size (in bytes)	Indicates the total size of directory entries currently present in the entry cache.
Maximum entry cache size (in bytes)	Indicates the size of the entry cache maintained by the directory. This value is managed by the “Maximum Cache Size” attribute. See “Tuning Database Performance,” on page 486, for information on changing this value using the Server Console.

Table 12-5 Database Performance Monitoring - Summary Information *(Continued)*

Performance Metric	Current Total
Current entry cache size (in entries)	Indicates the total number of directory entries currently present in the entry cache.
Maximum entry cache size (in entries)	Indicates the maximum number of directory entries that can be maintained in the entry cache. This value is managed by the "Maximum Entries in Cache" attribute. See "Tuning Database Performance," on page 486, for information on changing this value using the Server Console.

Database Cache Information Table

The Database Cache Information table provides caching information listed in Table 12-6.

Table 12-6 Database Performance Monitoring - Database Cache Information

Performance Metric	Current Total
Hits	Indicates the number of times the database cache successfully supplied a requested page. A page is a buffer of the size 2K.
Tries	Indicates the number of times the database cache was asked for a page.
Hit ratio	Indicates the ratio of database cache hits to database cache tries. The closer this value is to 100%, the better. Whenever a directory operation attempts to find a portion of the database that is not present in the database cache, the directory has to perform a disk access to obtain the appropriate database page. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory performance drops. To improve this ratio, you can increase the amount of data that the directory maintains in the database cache by increasing the value of the "Maximum Cache Size" attribute. See "Tuning Database Performance," on page 486, for information on changing this value using the Server Console.
Pages read in	Indicates the number of pages read from disk into the database cache.
Pages written out	Indicates the number of pages written from the cache back to disk. A database page is written to disk whenever a read-write page has been modified and then subsequently deleted from the cache. Pages are deleted from the database cache when the cache is full and a directory operation requires a database page that is not currently stored in cache.
Read-only page evicts	Indicates the number of read-only pages discarded from the cache to make room for new pages.

Table 12-6 Database Performance Monitoring - Database Cache Information *(Continued)*

Performance Metric	**Current Total**
Read-write page evicts	Indicates the number of read-write pages discarded from the cache to make room for new pages. This value differs from Pages Written Out in that these are discarded read-write pages that have not been modified.

Database File-Specific Table

The directory displays a table for each index file that makes up your database. Each of the tables provides the following information:

Table 12-7 Database Performance Monitoring - Database File-Specific

Performance Metric	**Current Total**
Cache hits	Number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
Cache misses	Number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
Pages read in	Indicates the number of pages brought to the cache from this file.
Pages written out	Indicates the number of pages for this file written from cache to disk.

Monitoring Databases from the Command-Line

You can monitor your directory's database activities from any LDAP client by performing a search operation with the following characteristics:

- Search for attribute `objectClass=*`
- Search base: `cn=monitor,cn=`*database_instance*`,cn=ldbm database, cn=plugins, cn=config,` where *database_instance* is the name of the database that you want to monitor
- Search scope: `base`

For example:

```
ldapsearch -h directory.example.com -s base -b
"cn=monitor,cn=Example,cn=ldbm database,cn=plugins, cn=config"
"objectclass=*"
```

In this example, the ldapsearch operation looks for the `Example` database. For information on searching the directory, see "Using ldapsearch," on page 590.

When you monitor your server's activities, you see the following information:

- `database`: Identifies the type of database you are currently monitoring.
- `readonly`: Indicates whether the database is in read-only mode. `0` indicates that the server is not in read-only mode, `1` indicates that it is in read-only mode.
- `entrycachehits`: Provides the same information as described in Entry cache hits in Table 12-5, on page 469.
- `entrycachetries`: Provides the same information as described in Entry cache tries in Table 12-5, on page 469.
- `entrycachehitratio`: Provides the same information as described in Entry cache hit ratio in Table 12-5, on page 469.
- `currententrycachesize`: Provides the same information as described in Current entry cache size (in entries) in Table 12-5, on page 469.
- `maxentrycachesize`: Provides the same information as described in Maximum entry cache size (in entries) in Table 12-5, on page 469.
- `dbchehits`: Provides the same information as described in Hits in Table 12-6, on page 470.
- `dbcachetries`: Provides the same information as described in Tries in Table 12-6, on page 470.
- `dbcachehitratio`: Provides the same information as described in Hit ratio in Table 12-6, on page 470.
- `dbcachepagein`: Provides the same information as described in Pages read in in Table 12-6, on page 470.
- `dbcachepageout`: Provides the same information as described in Pages written out in Table 12-6, on page 470.
- `dbcacheroevict`: Provides the same information as described in Read-only page evicts in Table 12-6, on page 470.
- `dbcacherwevict`: Provides the same information as described in Read-write page evicts in Table 12-6, on page 470.

Next, the following information for each file that makes up your database is displayed:

- `dbfilename-`*number*: Indicates the name of the file. *number* provides a sequential integer identifier (starting at 0) for the file. All associated statistics for the file are given this same numerical identifier.
- `dbfilecachehit-`*number*: Provides the same information as described in Cache hits in Table 12-7, on page 471.
- `dbfilecachemiss-`*number*: Provides the same information as described in Cache misses in Table 12-7, on page 471.
- `dbfilepagein-`*number*: Provides the same information as described in Pages read in in Table 12-7, on page 471.
- `dbfilepageout-`*number*: Provides the same information as described in Pages written out in Table 12-7, on page 471.

Monitoring Database Link Activity

You can monitor the activity of your database links from the command-line using the monitoring attributes. Use the `ldapsearch` command-line utility to return the attribute values that interest you. The monitoring attributes are stored in the following entry: `cn=monitor,cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config`.

For example, you can use the `ldapsearch` command-line utility to retrieve the number of add operations received by a particular database link called DBLink1. First, type the following to change to the directory containing the utility:

```
cd serverRoot/shared/bin
```

Then, run `ldapsearch` as follows:

```
ldapsearch -h directory.example.com -p 389 -D "cn=Directory
Manager" -w secret -s sub -b "cn=monitor,cn=DBLink1,cn=chaining
database,cn=plugins,cn=config" "(objectclass=*)" nsAddCount
```

NOTE The above command should be typed on a single line. It does not appear on one line here because of page size constraints.

You can search for the following database link monitoring attributes:

Table 12-8 Database Link Monitoring Attributes

Attribute Name	Description
`nsAddCount`	Number of add operations received.
`nsDeleteCount`	Number of delete operations received.
`nsModifyCount`	Number of modify operations received.
`nsRenameCount`	Number of rename operations received.
`nsSearchBaseCount`	Number of base level searches received.
`nsSearchOneLevelCount`	Number of one-level searches received.
`nsSearchSubtreeCount`	Number of subtree searches received.
`nsAbandonCount`	Number of abandon operations received.
`nsBindCount`	Number of bind request received.
`nsUnbindCount`	Number of unbinds received.
`nsCompareCount`	Number of compare operations received.
`nsOperationConnectionCount`	Number of open connections for normal operations.
`nsBindConnectionCount`	Number of open connections for bind operations.

For more information about `ldapsearch`, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

Chapter 13

Monitoring Directory Server Using SNMP

The server and database activity monitoring log setup described in chapter 12, “Monitoring Server and Database Activity,” is specific to Red Hat Directory Server (Directory Server). You can also monitor your Directory Server using the Simple Network Management Protocol (SNMP), which is a management protocol used for monitoring network activity which can be used to monitor a wide range of devices in real time.

SNMP has become interoperable on account of its widespread popularity. It is this interoperability, combined with the fact that SNMP can take on numerous jobs specific to a whole range of different device classes, that make SNMP the ideal standard mechanism for global network control and monitoring. SNMP allows network administrators to unify all network monitoring activities, with Directory Server monitoring part of the broader picture.

SNMP statistic reporting is available via a Net-SNMP subagent. The subagent will send *traps*, as well as report various statistics about your monitored Directory Server instances.

This chapter contains the following sections:

About SNMP

SNMP is a protocol used to exchange data about network activity. With SNMP, data travels between a managed device and a network management application (NMS) where users remotely manage the network. A managed device is anything that runs SNMP, such as hosts, routers, and your Directory Server. An NMS is usually a powerful workstation with one or more network management applications installed. A network management application graphically shows information about managed devices, which device is up or down, which and how many error messages were received, and so on.

Information is transferred between the NMS and the managed device through the use of two types of agents: the subagent and the master agent. The subagent gathers information about the managed device and passes the information to the master agent. Directory Server has a subagent. The master agent exchanges information between the various subagents and the NMS. The master agent runs on the same host machine as the subagents it talks to.

You can have multiple subagents installed on a host machine. For example, if you have Directory Server and a messaging server all installed on the same host, the subagents for both of these servers communicate with the same master agent.

Values for SNMP attributes, otherwise known as variables, that can be queried are kept on the managed device and reported to the NMS as necessary. Each variable is known as a managed object, which is anything the agent can access and send to the NMS. All managed objects are defined in a management information base (MIB), which is a database with a tree-like hierarchy. The top level of the hierarchy contains the most general information about the network. Each branch underneath is more specific and deals with separate network areas.

SNMP exchanges network information in the form of *protocol data unit* (PDUs). PDUs contain information about variables stored on the managed device. These variables, also known as managed objects, have values and titles that are reported to the NMS as necessary. Communication between an NMS and a managed device takes place either by the NMS sending updates or requesting information or by the managed object sending a notice or warning, called a *trap*, when a server shuts down or starts up.

Configuring the Master Agent

To use the subagent, you must have a Net-SNMP v5.21 master agent running on your system. You can download the Net-SNMP Master Agent from the Net-SNMP website (`http://www.net-snmp.org`).

The SNMP subagent included with Directory Server uses the AgentX protocol to communicate with the SNMP master agent running on your system. You must make sure that you enable AgentX support on your master agent. This is typically done by adding a line containing `agentx master` in the master agent's `snmpd.conf` file. For more details on configuring the master agent for AgentX support, refer to the Net-SNMP website (`http://www.net-snmp.org`).

Configuring the Subagent

The Directory Server SNMP subagent is located in *server_root*`/bin/slapd/server/ldap-agent`.

Subagent Configuration File

To use your subagent, you must first create a subagent configuration file. You can name this file whatever you like, and place it wherever you like. This configuration file is used to specify how to communicate with your master agent, logfile location, and which Directory Server instances to monitor.

agentx-master

The `agentx-master` setting tells the subagent how to communicate with the SNMP master agent. If this setting is not specified, the subagent will try to communicate the the master agent via the Unix domain socket `/var/agentx/master`. This is also where the Net-SNMP master agent listens for AgentX communications by default. If you configured your master agent to listen on a different Unix domain socket, you must use the agentx-master setting for your subagent to communicate with your master agent. If your master agent were listening on `/var/snmp/agenx`, the `agentx-master` setting would be `agentx-master /var/snmp/agentx`. Make sure that the user as whom you are running the subagent has the appropriate permissions to write to this socket.

If the master agent is listening for AgentX communications on a TCP port, you would have an `agentx-master` setting of `agentx-master localhost:705`.

agentx-logdir

The `agent-logdir` setting specifies the directory where you want the subagent to write its logfile. For example:

```
agentx-logdir /var/log
```

If this parameter is not specified, the agent will write its logfile to the same location as your subagent configuration file. The logfile will be named `ldap-agent.log`.

Make sure that the user as whom your subagent is running has write permission to this directory.

server

The server setting specifes a Directory Server instance that you want to monitor. You must use one server setting for each Directory Server instance. The subagent requires at least one server setting to be specified in its configuration file. The server setting should be set to the absolute path to the log directory of the Directory Server instance you would like to monitor. For example:

```
server /opt/redhat-ds/slapd-phonebook/logs
```

Make sure that the user as whom you are running your subagent has read permission to this directory.

Starting the Subagent

Once your master agent is running and you have created your subagent configuration file, you are ready to start the subagent. To start your subagent, you must run the `ldap-agent` program, specifying your subagent configuration file as an argument. You must supply the absolute path to the configuration file:

```
./ldapagent /opt/redhat-ds/ldap-agent.conf
```

If you want to enable extra debug logging, you can specify the `-D` option during startup:

```
./ldapagent -D /opt/redhat-ds/ldap-agent.conf
```

To stop your subagent, you must use the kill command against its process ID. Your subagent will print its process ID in its logfile, or you can run `ps -ef | grep ldap-agent` to find the process ID.

Testing the Subagent

To test your subagent, you can use the Net-SNMP toolkit command-line utilities, such as `snmpwalk` and `snmpget`. In order for these tools to display variable names for the Directory Server, you must configure them to load the Directory Server's MIB file. The Directory Server's MIB file is located in *server_root*`/plugins/snmp` directory. There are some additional common required MIB files located in *server_root*`/plugins/snmp/mibs` if you don't already have them. For details on configuring and using the Net-SNMP command-line tools, refer to the Net-SNMP website, `http://www.net-snmp.org`.

Configuring the Directory Server for SNMP

By default, the SNMP statics collection is enabled in the Directory Server. Using the Console, you can disable SNMP for a Directory Server instance or add SNMP management information. To configure SNMP settings from the Directory Server Console:

1. Select the Configuration tab, and then select the topmost entry in the navigation tree in the left pane.
2. Select the SNMP tab in the right pane.
3. Select the "Enable Statistics Collection" checkbox to enable Directory Server statistics collection. Clear the checkbox to disable it.
4. Enter a description that uniquely describes the directory instance in the Description text box.
5. Type the name the company or organization to which the directory belongs in the Organization text box.
6. Type the location within the company or organization where the directory resides in the Location text box.
7. Type the email address of the person responsible for maintaining the directory in the Contact text box.
8. Click Save.

Using the Management Information Base

The Directory Server's MIB is a file called `redhat-directory.mib`. This MIB contains definitions for variables pertaining to network management for the directory. These variables are known as managed objects. Using the directory MIB and Net-SNMP, you can monitor your directory like all other managed devices on your network. For more information on using the MIB, see "Testing the Subagent," on page 478.

You can see administrative information about your directory and monitor the server in real-time using the directory MIB. The directory MIB is broken into three distinct tables of managed objects:

- Operations Table
- Entries Table
- Interaction Table

NOTE Before you can use the directory's MIB, you must compile it along with the MIBs that you will find in the default location *serverRoot*`/plugins/snmp/mibs`

Operations Table

The Operations Table provides statistical information about Directory Server access, operations, and errors. describes the managed objects stored in the Operations Table of the `redhat-directory.mib` file.

Table 13-1 Operations - Managed Objects and Descriptions

Managed Object	Description
`dsAnonymousBinds`	The number of anonymous binds to the directory since server startup.
`dsUnauthBinds`	The number of unauthenticated binds to the directory since server startup.
`dsSimpleAuthBinds`	The number of binds to the directory that were established using a simple authentication method (such as password protection) since server startup.
`dsStrongAuthBinds`	The number of binds to the directory that were established using a strong authentication method (such as SSL or a SASL mechanism like Kerberos) since server startup.
`dsBindSecurityErrors`	The number of bind requests that have been rejected by the directory due to authentication failures or invalid credentials since server startup.
`dsInOps`	The number of operations forwarded to this directory from another directory since server startup.
`dsReadOps`	The number of read operations serviced by this directory since application start. The value of this object will always be `0` because LDAP implements read operations indirectly via the search operation.

Table 13-1 Operations - Managed Objects and Descriptions *(Continued)*

Managed Object	Description
`dsCompareOps`	The number of compare operations serviced by this directory since server startup.
`dsAddEntryOps`	The number of add operations serviced by this directory since server startup.
`dsRemoveEntryOps`	The number of delete operations serviced by this directory since server startup.
`dsModifyEntryOps`	The number of modify operations serviced by this directory since server startup.
`dsModifyRDNOps`	The number of modify RDN operations serviced by this directory since server startup.
`dsListOps`	The number of list operations serviced by this directory since server startup. The value of this object will always be `0` because LDAP implements list operations indirectly via the search operation.
`dsSearchOps`	The total number of search operations serviced by this directory since server startup.
`dsOneLevelSearchOps`	The number of one-level search operations serviced by this directory since server startup.
`dsWholeSubtreeSearchOps`	The number of whole subtree search operations serviced by this directory since server startup.
`dsReferrals`	The number of referrals returned by this directory in response to client requests since server startup.
`dsSecurityErrors`	The number of operations forwarded to this directory that did not meet security requirements.
`dsErrors`	The number of requests that could not be serviced due to errors (other than security or referral errors). Errors include name errors, update errors, attribute errors, and service errors. Partially serviced requests will not be counted as an error.

Entries Table

The Entries Table provides information about the contents of the directory entries. Table 13-2 describes the managed objects stored in the Entries Table in the `redhat-directory.mib` file.

Table 13-2 Entries - Managed Objects and Descriptions

Managed Object	Description
`dsMasterEntries`	The number of directory entries for which this directory contains the master entry. The value of this object will always be `0` (as no updates are currently performed).
`dsCopyEntries`	The number of directory entries for which this directory contains a copy. The value of this object will always be `0` (as no updates are currently performed).
`dsCacheEntries`	The number of entries cached in the directory.
`dsCacheHits`	The number of operations serviced from the locally held cache since application startup.
`dsSlaveHits`	The number of operations that were serviced from locally held replications (shadow entries). The value of this object will always be `0`.

Interaction Table

The Interaction Table provides statistical information about the interaction of this Directory Server with peer Directory Servers. This table:

- Contains statistical information for the last five Directory Servers with which this Directory Server has attempted to communicate.
- Provides useful information about how the interaction with peer Directory Servers affects the performance of this Directory Server.

Table 13-3 describes the managed objects stored in the Interaction Table of the `redhat-directory.mib` file.

Table 13-3 Interaction - Managed Objects and Descriptions

Managed Object	Description
`dsIntTable`	Each row of this table contains some details related to the history of the interaction of the monitored Directory Servers with their respective peer Directory Servers.
`dsIntEntry`	The entry containing interaction details of a Directory Server with a peer Directory Server.

Table 13-3 Interaction - Managed Objects and Descriptions *(Continued)*

Managed Object	Description
`dsIntIndex`	Together with `applIndex`, it forms the unique key to identify the conceptual row which contains useful information on the (attempted) interaction between the Directory Server (referred to by `applIndex`) and a peer Directory Server.
`dsName`	The distinguished name (DN) of the peer Directory Server to which this entry belongs.
`dsTimeOfCreation`	The value of `sysUpTime` when this row was created. If the entry was created before the network management subsystem was initialized, this object will contain a value of zero.
`dsTimeOfLastAttempt`	The value of `sysUpTime` when the last attempt was made to contact this Directory Server. If the last attempt was made before the network management subsystem was initialized, this object will contain a value of zero.
`dsTimeOfLastSuccess`	The value of `sysUpTime` when the last attempt made to contact this Directory Server was successful. This entry will have a value of zero if there have been no successful attempts or if the last successful attempt was made before the network management subsystem was initialized.
`dsFailuresSinceLastSuccess`	The number of failures since the last time an attempt to contact this Directory Server was successful. If there has been no successful attempts, this counter will contain the number of failures since this entry was created.
`dsFailures`	Cumulative failures since the creation of this entry.
`dsSuccesses`	Cumulative successes since the creation of this entry.
`dsURL`	The URL of the Directory Server application.

Chapter 14

Tuning Directory Server Performance

This chapter describes the tools provided with Red Hat Directory Server (Directory Server) to help optimize performance. It also provides tips to improve the performance of your directory.

This chapter contains the following sections:

Tuning Server Performance

You can manage your server's performance by limiting the amount of resources the server uses to proces client search requests. You can define:

- The maximum number of entries the server returns to the client in response to a search operation (size limit attribute).
- The maximum amount of real time (in seconds) you want the server to spend performing a search request (time limit attribute).
- The time (in seconds) during which the server maintains an idle connection before terminating it (idle timeout attribute).
- The maximum number of file descriptors available to the Directory Server (max number of file descriptors attribute).

To configure Directory Server to optimize performance:

1. In the Directory Server Console, select the Configuration tab, and then select the topmost entry in the navigation tree in the left pane.

   The tabs that are displayed in the right pane control server-wide configuration attributes.

2. Select the Performance tab in the right pane.

   The current server performance settings appear.

3. Set the maximum number of entries the server will return to the client in response to a search operation by entering a new value in the Size Limit text box.

   If you do not want to set a limit, type `-1` in this text box.

4. Enter the maximum amount of real time (in seconds) you want the server to spend performing a search request in the Time Limit text box.

   If you do not want to set a limit, type zero (`0`) in this text box.

5. Enter the time (in seconds) during which you want the server to maintain an idle connection before terminating it in the Idle Timeout text box.

   If you do not want to set a limit, type zero (`0`) in this text box.

6. Set the maximum number of file descriptors available to the Directory Server in the Max Number of File Descriptors text box. For more information on this parameter, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

For a better understanding of how these parameters impact your server's search performance, see "About Indexes," on page 391.

Tuning Database Performance

This section is divided into the following parts which describe methods for tuning database performance:

- Optimizing Search Performance
- Tuning Transaction Logging
- Changing the Location of the Database Transaction Log
- Changing the Database Checkpoint Interval

- Disabling Durable Transactions
- Specifying Transaction Batching

Optimizing Search Performance

You can improve server performance on searches by tuning database settings. The database attributes that affect performance mainly define the amount of memory available to the server.

To improve the cache hit ratio on search operations, you can increase the amount of data that the Directory Server maintains in the database cache. Do this by increasing the number of entries stored in the cache and by increasing the cache size. The maximum values that you can set for these attributes depends on the amount of real memory on your machine. Roughly, the amount of available memory on your machine should always be greater than `Maximum Entries in Cache x Average Entry Size.`

Use caution when changing these two attributes. Your ability to improve server performance with these attributes depends on the size of your database, the amount of physical memory available on your machine, and whether directory searches are random (that is, if your directory clients are searching for random and widely scattered directory data).

If your database does not fit into memory and if searches are random, attempting to increase the values set on these attributes does not help directory performance. In fact, changing these attributes may harm overall performance.

You can tune the following attributes:

- The attributes of the database that manages all other database instances. In Directory Server Console, you can see only the databases that contain your directory data and the `NetscapeRoot` database. However, the server uses another database to manage these. On this database, you can change the following attributes to improve performance:
    - The amount of memory that you want to make available for all databases (maximum cache size attribute).
    - The maximum number of entries you want the server to verify in response to a search request (look-through limit attribute).
- The attributes of each database that you use to store directory data, including the server configuration data in the `NetscapeRoot` database. On these databases, you can change the following attributes to improve performance:

- ❍ The maximum number of entries you want the server to keep in memory (maximum entries in cache attribute).
- ❍ The amount of memory you want to make available for cached entries (memory available for cache attribute).

To configure the default database attributes that apply to all other database instances:

1. In the Directory Server Console, select the Configuration tab; then, in the navigation tree, expand the Data Icon, and highlight the Database Settings node.

   This displays the Database tabs in the right pane.

2. Select the LDBM Plugin Settings tab in the right pane.

   This tab contains the database attributes for all databases stored on this server.

3. In the Maximum Cache Size field, enter a value corresponding to the amount of memory that you want to make available for all databases.
4. In the look-through limit field, enter the maximum number of entries you want the server to check in response to a search request.

   If you do not want to set a limit, type `-1` in this text box. If you bind to the directory as the Directory Manager, by default the look-through limit is unlimited and overrides any settings you specify here.

To configure the attributes of each database that stores your directory data:

1. In the Directory Server Console, select the Configuration tab; then, in the navigation tree, expand the Data Icon. Expand the suffix of the database you want to tune, and highlight the database.

   The tabs displayed in the right pane control parameter settings for this database.

2. Select the Database Settings tab in the right pane.
3. Enter the number of entries you want the server to keep in memory in the Maximum Entries in Cache field.

4. Enter the amount of memory you want to make available for cached entries in the Memory Available for Cache field.

If you are creating a very large database from LDIF, set this attribute as large as possible, depending on the memory available on your machine. The larger this parameter, the faster your database will be created.

When you have finished creating your database, be sure to set this parameter back to some lower value before you run your server in a production environment.

Tuning Transaction Logging

Every Directory Server contains a transaction log which logs operations for all the databases it manages. Whenever a directory database operation such as a write is performed, the server logs the operation to the transaction log. For best performance, the directory does not perform the operation immediately. Instead, the operation is stored in a temporary memory cache on the Directory Server until the operation is completed.

If the server experiences a failure, such as a power outage, and shuts down abnormally, the information about recent directory changes that were stored in the cache is lost. However, when the server restarts, the directory automatically detects the error condition and uses the database transaction log to recover the database.

Although database transaction logging and database recovery are automatic processes that require no intervention, you may want to tune some of the database transaction logging attributes to optimize performance.

CAUTION The transaction logging attributes are provided only for system modifications and diagnostics. These settings should be changed only with the guidance of Red Hat Professional Services or Red Hat Technical Support.

Setting these attributes and other configuration attributes inconsistently may cause the directory to be unstable.

Changing the Location of the Database Transaction Log

By default, the database transaction log file is stored in the *serverRoot*`/slapd-`*serverID*`/db` directory along with the database files themselves. Because the purpose of the transaction log is to aid in the recovery of a directory database that was shut down abnormally, it is a good idea to store the database transaction log on a different disk from the one containing the directory database. Storing the database transaction log on a separate physical disk may also improve directory performance.

To change the location of the database transaction log file, use the following procedure:

1. Stop the Directory Server.

   For instructions, refer to "Starting and Stopping the Server from the Console," on page 38.

2. Use the `ldapmodify` command-line utility to add the `nsslapd-db-logdirectory` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry. Provide the full path to the log directory in the attribute.

   For information on the `nsslapd-db-logdirectory` attribute syntax, see the *Red Hat Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 60.

3. Restart Directory Server.

Changing the Database Checkpoint Interval

At regular intervals, the Directory Server writes operations logged in the transaction log to the disk and logs a checkpoint entry in the database transaction log. By indicating which changes have already been written to the directory, checkpoint entries indicate where to begin recovery from the transaction log, thus speeding up the recovery process.

By default, the Directory Server is set up to send a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval may increase the performance of directory write operations. However, increasing the checkpoint interval may also increase the amount of time required to recover directory databases after a disorderly shutdown and require more disk space due to large database transaction log files. Therefore, you should only modify this attribute if you are familiar with database optimization and can fully assess the effect of the change.

To modify the checkpoint interval while the server is running, use the `ldapmodify` command-line utility to add the `nsslapd-db-checkpoint-interval` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

For more information on the syntax of the `nsslapd-db-checkpoint-interval` attribute, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 60.

Disabling Durable Transactions

Durable transaction logging means that the temporary database transaction log is, in fact, physically written to disk.

When durable transaction logging is disabled, every directory database operation is written to the database transaction log file but may not be physically written to disk immediately. If a directory change was written to the logical database transaction log file but not physically written to disk at the time of a system crash, you cannot recover the change. When durable transactions are disabled, the recovered database is consistent but does not reflect the results of any LDAP write operations that completed just before the system crash.

By default, durable database transaction logging is enabled. To disable durable transaction logging, use the following procedure:

1. Stop the Directory Server.

   For instructions, refer to "Starting and Stopping the Server from the Command-Line," on page 38.

2. Use the `ldapmodify` command-line utility to add the `nsslapd-db-durable-transactions` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry, and set the value of this attribute to `off`.

   For information on the syntax of the `nsslapd-db-durable-transactions` attribute, see the *Red Hat Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 60.

3. Restart the Directory Server.

Specifying Transaction Batching

To improve update performance when full transaction durability is not required, you can use the `nsslapd-db-transaction-batch-val` attribute to specify how many transactions will be batched before being committed to the transaction log. Setting this attribute to a value of greater than `0` causes the server to delay committing transactions until the number of queued transactions is equal to the attribute value. For transaction batching to be valid, the `nsslapd-db-durable-transaction` attribute must be set to `on`.

To specify or modify transaction batching while the server is running, use the `ldapmodify` command-line utility to add the `nsslapd-db-transaction-batch-val` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

For more information on the syntax and values of the `nsslapd-db-transaction-batch-val` attribute, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 60.

Miscellaneous Tuning Tips

This section provides you with some performance related tips and concepts you ought to remember.

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 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`, as this allows you to centralize configuration information.

Part 2

Plug-ins Reference

Chapter 15

Administering Directory Server Plug-ins

Red Hat Directory Server (Directory Server) plug-ins extend the functionality of the server. Directory Server ships with several plug-ins to help you manage your directory. This chapter contains general information on the types of plug-ins available and how to enable or disable them. This chapter is divided into the following sections:

- Server Plug-in Functionality Reference (page 495)
- Enabling and Disabling Plug-ins from the Server Console (page 516)

Server Plug-in Functionality Reference

The following tables provide you with a quick overview of the plug-ins provided with Directory Server, along with their configurable options, configurable arguments, default setting, dependencies, general performance-related information, and further reading. These tables will allow you to weigh up plug-in performance gains and costs and choose the optimal settings for your deployment. The Further Information heading cross-references further reading, where this is available.

7-bit Check Plug-in

Table 15-1 Details of 7-Bit Check Plug-in

Plug-in Name	7-bit check (NS7bitAtt)

Table 15-1 Details of 7-Bit Check Plug-in *(Continued)*

DN of Configuration Entry	`cn=7-bit check,cn=plugins,cn=config`
Description	Checks certain attributes are 7-bit clean
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	List of attributes (`uid mail userpassword`) followed by "," and then suffix(es) on which the check is to occur.
Dependencies	None
Performance Related Information	None
Further Information	If your Directory Server uses non-ASCII characters, Japanese, for example, turn this plug-in off.

ACL Plug-in

Table 15-2 Details of ACI Plug-in

Plug-in Name	ACL Plug-in
DN of Configuration Entry	`cn=ACL Plugin,cn=plugins,cn=config`
Description	ACL access check plug-in
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	N/A
Further Information	Chapter 6, "Managing Access Control."

ACL Preoperation Plug-in

Table 15-3 Details of Preoperation Plug-in

Plug-in Name	ACL Preoperation
DN of Configuration Entry	`cn=ACL preoperation,cn=plugins,cn=config`
Description	ACL access check plug-in
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	database
Performance Related Information	None
Further Information	Chapter 6, "Managing Access Control."

Binary Syntax Plug-in

Table 15-4 Details of Binary Syntax Plug-in

Plug-in Name	Binary Syntax
DN of Configuration Entry	`cn=Binary Syntax,cn=plugins,cn=config`
Description	Syntax for handling binary data
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Boolean Syntax Plug-in

Table 15-5 Details of Boolean Syntax Plug-in

Plug-in Name	Boolean Syntax
DN of Configuration Entry	`cn=Boolean Syntax,cn=plugins,cn=config`
Description	Syntax for handling booleans
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Case Exact String Syntax Plug-in

Table 15-6 Details of Case Exact String Syntax Plug-in

Plug-in Name	Case Exact String Syntax
DN of Configuration Entry	`cn=Case Exact String Syntax,cn=plugins,cn=config`
Description	Syntax for handling case-sensitive strings
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Case Ignore String Syntax Plug-in

Table 15-7 Details of Case Ignore String Syntax Plug-in

Plug-in Name	Case Ignore String Syntax
DN of Configuration Entry	`cn=Case Ignore String Syntax,cn=plugins,cn=config`
Description	Syntax for handling case-insensitive strings
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Chaining Database Plug-in

Table 15-8 Details of Cloning Database Plug-in

Plug-in Name	Chaining Database
DN of Configuration Entry	`cn=Chaining database,cn=plugins,cn=config`
Description	Syntax for handling DNs
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 3, "Configuring Directory Databases."

Class of Service Plug-in

Table 15-9 Details of Class of Service Plug-in

Plug-in Name	Class of Service
DN of Configuration Entry	`cn=Class of Service,cn=plugins,cn=config`
Description	Allows for sharing of attributes between entries
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 5, "Advanced Entry Management."

Country String Syntax Plug-in

Table 15-10 Details of Country String Plug-in

Plug-in Name	Country String Syntax Plug-in
DN of Configuration Entry	`cn=Country String Syntax,cn=plugins,cn=config`
Description	Syntax for handling countries
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Distinguished Name Syntax Plug-in

Table 15-11 Details of Distinguished Name Syntax Plug-in

Plug-in Name	Distinguished Name Syntax
DN of Configuration Entry	`cn=Distinguished Name Syntax,cn=plugins,cn=config`
Description	Syntax for handling DNs
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Generalized Time Syntax Plug-in

Table 15-12 Details of Generalized Time Syntax Plug-in

Plug-in Name	Generalized Time Syntax
DN of Configuration Entry	`cn=Generalized Time Syntax,cn=plugins,cn=config`
Description	Syntax for dealing with dates, times and time zones
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.

Table 15-12 Details of Generalized Time Syntax Plug-in *(Continued)*

Further Information	The Generalized Time String consists of the following: • four digit year • two digit month (for example, 01 for January) • two digit day, two digit hour • two digit minute • two digit second • decimal part of a second (*optional*) • a time zone indication We strongly recommend that you use the Z time zone indication, which stands for Greenwich Mean Time.

Integer Syntax Plug-in

Table 15-13 Details of Integer Syntax Plug-in

Plug-in Name	Integer Syntax
DN of Configuration Entry	`cn=Integer Syntax,cn=plugins,cn=config`
Description	Syntax for handling integers
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Internationalization Plug-in

Table 15-14 Details of Internationalization Plug-in

Plug-in Name	Internationalization Plug-in
DN of Configuration Entry	`cn=Internationalization Plugin,cn=plugins,cn=config`
Description	Syntax for handling international characters (in DNs)
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	The Internationalization Plug-in has one argument which must not be modified: *serverRoot*`/slapd-`*serverID*`/config/slapd-collations.conf` This directory stores the collation orders and locales used by the Internationalization Plug-in.
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	See Appendix D, "Internationalization."

ldbm Database Plug-in

Table 15-15 Details of ldbm Database Plug-in

Plug-in Name	ldbm database Plug-in
DN of Configuration Entry	`cn=ldbm database plug-in,cn=plugins,cn=config`
Description	Implements local databases
Configurable Options	N/A
Default Setting	`on`
Configurable Arguments	None
Dependencies	None

Table 15-15 Details of ldbm Database Plug-in *(Continued)*

Performance Related Information	See *Red Hat Directory Server Configuration, Command, and File Reference* for further information on ldbm database plug-in attributes.
Further Information	Chapter 3, "Configuring Directory Databases."

Legacy Replication Plug-in

Table 15-16 Details of Legacy Replication Plug-in

Plug-in Name	Legacy Replication Plug-in
DN of Configuration Entry	`cn=Legacy Replication plug-in,cn=plugins,cn=config`
Description	Enables this version of Directory Server to be a consumer of a 4.x supplier
Configurable Options	`on` | `off`
Default Setting	`on`
Configurable Arguments	None. This plug-in can be disabled if the server is not (and never will be) a consumer of a 4.x server.
Dependencies	database
Performance Related Information	None
Further Information	Chapter 8, "Managing Replication."

Multi-Master Replication Plug-in

Table 15-17 Details of Multi-Master Replication Plug-in

Plug-in Name	Multi-master Replication Plug-in
DN of Configuration Entry	`cn=Multimaster Replication plugin,cn=plugins, cn=config`
Description	Enables replication between two Directory Servers
Configurable Options	`on` | `off`

Table 15-17 Details of Multi-Master Replication Plug-in *(Continued)*

Default Setting	`on`
Configurable Arguments	None
Dependencies	database
Performance Related Information	N/A
Further Information	You can turn this plug-in off if you only have one server, which will never replicate. See also Chapter 8, "Managing Replication."

Octet String Syntax Plug-in

Table 15-18 Details of Octet String Syntax Plug-in

Plug-in Name	Octet String Syntax
DN of Configuration Entry	`cn=Octet String Syntax,cn=plugins,cn=config`
Description	Syntax for handling octet strings
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

CLEAR Password Storage Plug-in

Table 15-19 Details of CLEAR Password Storage Plug-in

Plug-in Name	CLEAR
DN of Configuration Entry	`cn=CLEAR,cn=Password Storage Schemes,cn=plugins, cn=config`

Table 15-19 Details of CLEAR Password Storage Plug-in *(Continued)*

Description	CLEAR password storage scheme used for password encryption
Configurable Options	`on` | `off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 7, "User Account Management."

CRYPT Password Storage Plug-in

Table 15-20 Details of CRYPT Password Storage Plug-in

Plug-in Name	CRYPT
DN of Configuration Entry	`cn=CRYPT,cn=Password Storage Schemes,cn=plugins, cn=config`
Description	CRYPT password storage scheme used for password encryption
Configurable Options	`on` | `off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 7, "User Account Management."

NS-MTA-MD5 Password Storage Plug-in

Table 15-21 Details of NS-MTA-MD5 Password Storage Plug-in

Plug-in Name	NS-MTA-MD5
DN of Configuration Entry	`cn=NS-MTA-MD5,cn=Password Storage Schemes,cn=plugins, cn=config`
Description	NS-MTA-MD5 password storage scheme for password encryption
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Red Hat recommends that you leave this plug-in running at all times.
Further Information	You cannot choose to encrypt passwords using the NS-MTA-MD5 password storage scheme. The storage scheme is present in Red Hat Directory Server but only for reasons of backward compatibility with earlier versions of Directory Server. See Chapter 7, "User Account Management."

SHA Password Storage Plug-in

Table 15-22 Details of SHA Password Storage Plug-in

Plug-in Name	SHA
DN of Configuration Entry	`cn=SHA,cn=Password Storage Schemes,cn=plugins,cn=config`
Description	SHA password storage scheme for password encryption
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None

Table 15-22 Details of SHA Password Storage Plug-in *(Continued)*

Performance Related Information	If your directory does not contain passwords encrypted using the SHA password storage scheme, you may turn this plug-in off. SHA is only included for compatibility with earlier releases; it is recommended that you use SSHA rather than SHA because SSHA is a far more secure option.
Further Information	Chapter 7, "User Account Management."

SSHA Password Storage Plug-in

Table 15-23 Details of SSHA Password Storage Plug-in

Plug-in Name	SSHA
DN of Configuration Entry	`cn=SSHA,cn=Password Storage Schemes,cn=plugins,cn=config`
Description	SSHA password storage scheme for password encryption
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 7, "User Account Management."

Postal Address String Syntax Plug-in

Table 15-24 Details of Postal Address String Syntax Plug-in

Plug-in Name	Postal Address Syntax
DN of Configuration Entry	`cn=Postal Address Syntax,cn=plugins,cn=config`
Description	Syntax used for handling postal addresses

Table 15-24 Details of Postal Address String Syntax Plug-in *(Continued)*

Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

PTA Plug-in

Table 15-25 Details of PTA Plug-in

Plug-in Name	Pass-Through Authentication Plug-in
DN of Configuration Entry	`cn=Pass Through Authentication,cn=plugins,cn=config`
Description	Enables pass-through authentication, the mechanism which allows one directory to consult another to authenticate bind requests. This plug-in is not listed in Directory Server Console if you use the same server for your user directory and configuration directory.
Configurable Options	`on | off`
Default Setting	`off`
Configurable Arguments	`ldap://redhat.com:389/o=redhat`
Dependencies	None
Performance Related Information	Chapter 16, “Using the Pass-through Authentication Plug-in.”
Further Information	Chapter 16, “Using the Pass-through Authentication Plug-in.”

Referential Integrity Postoperation Plug-in

Table 15-26 Details of Referential Integrity Postoperation Plug-in

Plug-in Name	Referential Integrity Postoperation
DN of Configuration Entry	`cn=Referential Integrity Postoperation,cn=plugins, cn=config`
Description	Enables the server to ensure referential integrity
Configurable Options	`All` configuration and `on | off`
Default Setting	`off`
Configurable Arguments	When enabled, the postoperation Referential Integrity Plug-in performs integrity updates on the `member`, `uniquemember`, `owner` and `seeAlso` attributes immediately after a delete or rename operation. You can reconfigure the plug-in to perform integrity checks on all other attributes. Configurable arguments are as follows: 1. Check for referential integrity `-1` = no check for referential integrity `0` = check for referential integrity is performed immediately `positive integer` = request for referential integrity is queued and processed at a later stage. This positive integer serves as a wake-up call for the thread to process the request at intervals corresponding to the integer specified. 2. Log file for storing the change; for example `/opt/redhat-ds/logs/referint` 3. All the additional attrribute names you want to be checked for referential integrity.
Dependencies	database
Performance Related Information	You should enable the Referential Integrity Plug-in on only one master in a multimaster replication environment to avoid conflict resolution loops. When enabling the plug-in on chained servers, you must be sure to analyze your performance resource and time needs as well as your integrity needs.
Further Information	See “Maintaining Referential Integrity,” on page 75.

Retro Changelog Plug-in

Table 15-27 Details of Retro Changelog Plug-in

Plug-in Name	Retro Changelog Plug-in
DN of Configuration Entry	`cn=Retro Changelog Plugin,cn=plugins,cn=config`
Description	Used by LDAP clients for maintaining application compatibility with Directory Server 4.x versions. Maintains a log of all changes occuring in the Directory Server. The Retro Changelog offers the same functionality as the changelog in the 4.x versions of Directory Server.
Configurable Options	`on | off`
Default Setting	`off`
Configurable Arguments	See *Red Hat Directory Server Configuration, Command, and File Reference* for further information on the two configuration attributes for the Retro Changelog Plug-in.
Dependencies	None
Performance Related Information	May slow down Directory Server performance.
Further Information	Chapter 8, "Managing Replication."

Roles Plug-in

Table 15-28 Details of Roles Plug-in

Plug-in Name	Roles Plug-in
DN of Configuration Entry	`cn=Roles Plugin,cn=plugins,cn=config`
Description	Enables the use of roles in the Directory Server
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None

Table 15-28 Details of Roles Plug-in *(Continued)*

Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 5, "Advanced Entry Management."

Space Insensitive String Syntax Plug-in

Table 15-29 Details of Space Insensitive String Syntax Plug-in

Plug-in Name	Space Insensitive String Syntax
DN of Configuration Entry	`cn=Space Insensitive String Syntax,cn=plugins,cn=config`
Description	Syntax for handling space-insensitive values
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	This plug-in enables the Directory Server to support *space and case insensitive* values. Applications can now search the directory using entries with ASCII space characters. For example, a search or compare operation that uses `jOHN Doe` will match entries that contain `johndoe`, `john doe`, and `John Doe`. For more information about finding directory entries, see Appendix B, "Finding Directory Entries."

State Change Plug-in

Table 15-30 Details of State Change Plug-in

Plug-in Name	State Change Plug-in

Table 15-30 Details of State Change Plug-in *(Continued)*

DN of Configuration Entry	`cn=State Change Plugin,cn=plugins,cn=config`
Description	Enables state-change-notification service.
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	
Further Information	

Telephone Syntax Plug-in

Table 15-31 Details of Telephone Syntax Plug-in

Plug-in Name	Telephone Syntax
DN of Configuration Entry	`cn=Telephone Syntax,cn=plugins,cn=config`
Description	Syntax for handling telephone numbers
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

UID Uniqueness Plug-in

Table 15-32 Details of UID Uniqueness Plug-in

Plug-in Name	UID Uniqueness Plug-in
DN of Configuration Entry	`cn=UID Uniqueness,cn=plugins,cn=config`
Description	Checks that the values of specified attributes are unique each time a modification occurs on an entry.
Configurable Options	`on | off`
Default Setting	`off`

Configurable Arguments	Enter the following arguments:
	`uid`
	`"DN"`
	`"DN"...`
	if you want to check for uid attribute uniqueness in all listed subtrees.
	However, enter the following arguments:
	`attribute="uid"`
	`MarkerObjectclass = "ObjectClassName"`
	and optionally
	`requiredObjectClass = "ObjectClassName"`
	if you want to check for uid attribute uniqueness when adding or updating entries with the `requiredObjectClass`, starting from the parent entry containing the `ObjectClass` as defined by the `MarkerObjectClass` attribute.

Dependencies	N/A

Table 15-32 Details of UID Uniqueness Plug-in *(Continued)*

Performance Related Information	This plug-in may slow down Directory Server performance. In a multi-master replication environment, the UID Uniqueness Plug-in will not work at all and should therefore not be enabled. If you try to add a new entry to a server where the UID Uniqueness Plug-in is enabled and a referral has been created in a subtree, then the UID Uniqueness Plug-in will not work because if it sees any other error apart from `noSuchObject` (meaning that the entry does not already exist), which it will do if a referral is created, then it will return an operations error preventing you from adding your new entry. To prevent being blocked by such an operations error, disable the plug-in on the server where you created the referral. If, however, you still want to run a UID Uniqueness check, make sure that you only activate the plug-in on the last of the referred-to servers to prevent it from blocking the referral mechanism.
Further Information	Chapter 17, "Using the Attribute Uniqueness Plug-in."

URI Plug-in

Table 15-33 Details of URI Plug-in

Plug-in Name	URI Syntax
DN of Configuration Entry	`cn=URI Syntax,cn=plugins,cn=config`
Description	Syntax for handling URIs (Unique Resource Identifiers), including URLs (Unique Resource Locators)
Configurable Options	`on | off`
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Enabling and Disabling Plug-ins from the 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 checkbox. To enable the plug-in, check this checkbox.
5. Click Save.
6. Restart the Directory Server.

Chapter 16

Using the Pass-through Authentication Plug-in

Pass-through authentication (PTA) is a mechanism by which one Directory Server consults another to authenticate bind requests. The PTA Plug-in provides this functionality, allowing a Directory Server to accept simple bind operations (password-based) for entries not stored in its local database.

Red Hat Directory Server (Directory Server) uses PTA to allow you to administer your user and configuration directories on separate instances of Directory Server.

This chapter describes the PTA Plug-in in the following sections:

How Directory Server Uses PTA

If you install the configuration directory and the user directory on separate instances of Directory Server, the installation program automatically sets up PTA to allow the Configuration Administrator user (usually `admin`) to perform administrative duties.

PTA is required in this case because the `admin` user entry is stored under `o=NetscapeRoot` in the configuration directory. Therefore, attempts to bind to the user directory as `admin` would normally fail. PTA allows the user directory to transmit the credentials to the configuration directory, which verifies them. The user directory then allows the `admin` user to bind.

The user directory in this example acts as the PTA directory server, the server that passes through bind requests to another directory server. The configuration directory acts as the authenticating directory, the server that contains the entry and verifies the bind credentials of the requesting client.

You will also see the term *pass-through subtree* used in this chapter. The pass-through subtree is the subtree *not* present on the PTA directory. When a user's bind DN contains this subtree, the user's credentials are passed on to the authenticating directory.

NOTE The PTA Plug-in is not listed in Directory Server Console when you use the same server for your user directory and your configuration directory.

Here's how pass-through authentication works:

1. You install the configuration directory server (authenticating directory) on machine A.
   - Server name: `configdir.example.com`
   - Suffix: `o=NetscapeRoot`
2. You install the user directory server (PTA directory) on machine B.
   - Server name: `userdir.example.com`
   - Suffix: `dc=example,dc=com`
3. During the installation of the user directory on machine B, you are prompted to provide an LDAP URL. This URL points to the configuration directory on machine A.
4. The installation program adds an entry to the `dse.ldif` file on the user directory that enables the PTA Plug-in.

   This entry contains the LDAP URL you provided. For example:

   ```
   dn: cn=Pass Through Authentication,cn=plugins,
   objectClass: top
   objectClass: nsSlapdPlugin
   objectClass: extensibleObject
   cn: Pass Through Authentication
   nsslapd-pluginPath:
   /opt/redhat-ds/servers/lib/passthru-plugin.so
   nsslapd-pluginInitfunc: passthruauth_init
   nsslapd-pluginType: preoperation
   ```

```
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config.example.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

The user directory is now configured to send all bind requests for entries whose DN contains `o=NetscapeRoot` to the configuration directory `configdir.example.com`.

5. When installation is complete, the `admin` user attempts to connect to the user directory to begin adding users.
6. The setup program adds the `admin` user's entry to the directory as `uid=admin, ou=TopologyManagement,o=NetscapeRoot`. So the user directory passes the bind request through to the configuration directory as defined by the PTA Plug-in configuration.
7. The configuration directory authenticates the user's credentials and sends the information back to the user directory.
8. The user directory allows the `admin` user to bind.

PTA Plug-in Syntax

PTA Plug-in configuration information is specified in the `cn=Pass Through Authentication,cn=plugins,cn=config` entry in the `dse.ldif` file on the PTA directory (the user directory configured to pass through bind requests to the authenticating directory) using the syntax described in this section.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.extension
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: ldap|ldaps://authDS/subtree
[maxconns,maxops,timeout,ldver,connlifetime]
```

The variable components of the PTA plug-in syntax are described in Table 16-1.

NOTE

- The LDAP URL (`ldap|ldaps://`*`authDS`*`/`*`subtree`*) must be separated from the optional parameters (*maxconns*, *maxops*, *timeout*, *ldver*, *connlifetime*) by a single space.
- If you explicitly define any of the optional parameters, you must define all of them, even if you specify only the default values.
- You can specify several authenticating directories or subtrees by incrementing the `nsslapd-pluginarg` attribute suffix by 1 each time. This is illustrated in "Specifying Multiple Authenticating Directory Servers," on page 528.

The optional parameters are described in the following table in the order in which they appear in the syntax.

Table 16-1 PTA Plug-in Parameters

Variable	Definition
`state`	Defines whether the plug-in is enabled or disabled. Acceptable values are `on` or `off`. See "Turning the Plug-in On or Off," on page 522, for more information.
`extension`	File extension for the plug-in. The extension is always `.sl` on HP-UX and `.so` on all other UNIX platforms.
`ldap|ldaps`	Defines whether SSL is used for communication between the two Directory Servers. See "Configuring the Servers to Use a Secure Connection," on page 523, for more information.
`authDS`	The authenticating directory hostname. You can specify the port number of the directory server by adding a colon and then the port number. For example: `ldap://dirserver.example.com:390/` If you do not specify the port number, the PTA server attempts to connect using: • Port 389 if `ldap://` is specified in the URL. • Port 636 if `ldaps://` is specified in the URL. See "Specifying the Authenticating Directory Server," on page 524, for more information.
`subtree`	The pass-through subtree. The PTA directory server passes through bind requests to the authenticating directory server from all clients whose DN is in this subtree. See "Specifying the Pass-through Subtree," on page 525, for more information.

Table 16-1 PTA Plug-in Parameters *(Continued)*

Variable	Definition
`maxconns`	*Optional*. The maximum number of connections the PTA directory can simultaneously open to the authenticating directory. The default is `3`. See "Configuring the Optional Parameters," on page 525, for more information.
`maxops`	*Optional*. The maximum number of simultaneous operations (usually bind requests) the PTA directory can send to the authenticating directory within a single connection. The default is `5`. See "Configuring the Optional Parameters," on page 525, for more information.
`timeout`	*Optional*. The time limit, in seconds, that the PTA directory waits for a response from the authenticating directory server. If this timeout is exceeded, the server returns an error to the client. The default is `300` seconds (five minutes). Specify zero (`0`) to indicate no time limit should be enforced. See "Configuring the Optional Parameters," on page 525, for more information.
`ldver`	*Optional*. The version of the LDAP protocol used to connect to the authenticating directory. Directory Server supports LDAP version 2 and 3. The default is version 3. See "Configuring the Optional Parameters," on page 525, for more information.
`connlifetime`	*Optional*. The time limit, in seconds, within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating directory. The server will not close the connection unless a bind request is initiated and the directory determines the connection lifetime has been exceeded. If you do not specify this option, or if only one host is listed, no connection lifetime will be enforced. If two or more hosts are listed, the default is 300 seconds (five minutes). See "Configuring the Optional Parameters," on page 525, for more information.

Configuring the PTA Plug-in

The only method for configuring the PTA plug-in is to modify the entry `cn=Pass Through Authentication,cn=plugins,cn=config` in the `dse.ldif` file. To modify the `dse.ldif` file, you must proceed as follows:

1. Use the `ldapmodify` command to modify `cn=Pass Through Authentication,cn=plugins,cn=config.`

2. Restart Directory Server.

Before you configure any of the parameters discussed in this section, the PTA Plug-in entry must be present in the `dse.ldif` file. If this entry does not exist, you must create it with the appropriate syntax, as described in "PTA Plug-in Syntax," on page 519.

NOTE If you installed the user and configuration directories on different instances of the directory, the PTA Plug-in entry is automatically added to the user directory's `dse.ldif` file. If you installed the user and configuration directories on the same instance, the syntax is not automatically added, and you need to add it manually.

This section provides information about configuring the plug-in in the following sections:

- Turning the Plug-in On or Off
- Configuring the Servers to Use a Secure Connection
- Specifying the Authenticating Directory Server
- Specifying the Pass-through Subtree
- Configuring the Optional Parameters

Turning the Plug-in On or Off

To turn the PTA Plug-in on from the command-line:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

2. Use the `ldapmodify` command to import the LDIF file into the directory.

   For detailed information on the `ldapmodify` command, refer to *Red Hat Directory Server Configuration, Command, and File Reference.*

3. When you enable the plug-in, you must also check that the plug-in initialization function is properly defined.

The entry `cn=Pass Through Authentication,cn=plugins,cn=config` should contain the following attribute-value pairs:

```
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.extension
nsslapd-pluginInitfunc: passthruauth_init
```

where *extension* is always `.sl` on HP-UX and `.so` on all other UNIX platforms.

4. Restart the server.

For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

To disable the plug-in, change the LDIF update statements to delete the

```
nsslapd-pluginenabled: on
```

statement, and add the

```
nsslapd-pluginenabled: off
```

statement. Whenever you enable or disable the PTA Plug-in from the command-line, you must restart the server.

Configuring the Servers to Use a Secure Connection

You can configure the PTA directory to communicate with the authenticating directory over SSL. You do this by specifying LDAPS in the LDAP URL of the PTA directory.

To configure the PTA directory and authenticating directory to use SSL:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldaps://authDS/subtree [optional_parameters]
```

For information on the variable components in this sytax, refer to Table 16-1, on page 520.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

Specifying the Authenticating Directory Server

The authenticating directory contains the bind credentials for the entry with which the client is attempting to bind. The PTA directory passes the bind request to the host you define as the authenticating directory. You specify the authenticating directory server by replacing `authDS` in the LDAP URL of the PTA directory with the authenticating directory's hostname.

To specify the authenticating directory for PTA:

1. Create an LDIF file that contains the following LDIF update statements:

   ```
   dn: cn=Pass Through Authentication,cn=plugins,cn=config
   cn: Pass Through Authentication
   changetype: add
   add: nsslapd-pluginarg0
   nsslapd-pluginarg0: ldap://authDS/subtree [optional_parameters]
   ```

   Optionally, you can include a colon followed by a port number. If you do not specify the port number, the PTA directory server attempts to connect using:

   - Port 389 if `ldap://` is specified in the URL.
   - Port 636 if `ldaps://` is specified in the URL.

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

   ```
   "ldap://dirserver.example.com:389/subtree [Parameters]"
   ```

   For information on the variable components in this sytax, refer to Table 16-1, on page 520.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

Specifying the Pass-through Subtree

The PTA directory passes through bind requests to the authenticating directory from all clients whose DN is defined in the pass-through subtree. You specify the subtree by replacing the `subtree` parameter in the LDAP URL of the PTA directory.

The pass-through subtree must not exist in the PTA directory. If it does, the PTA directory attempts to resolve bind requests using its own directory contents and the binds fail.

To specify the pass-through subtree:

1. Create an LDIF file that contains the following LDIF update statements:

   ```
   dn: cn=Pass Through Authentication,cn=plugins,cn=config
   cn: Pass Through Authentication
   changetype: add
   add: nsslapd-pluginarg0
   nsslapd-pluginarg0: ldap://authDS/subtree [optional_parameters]
   ```

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

   ```
   "ldap://dirserver.example.com/o=NetscapeRoot [Parameters]"
   ```

   For information on the variable components in this sytax, refer to Table 16-1, on page 520.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

Configuring the Optional Parameters

You can configure the following optional parameters for the PTA Plug-in:

- The maximum number of connections the PTA directory server can open simultaneously to the authenticating directory, represented by `maxconns` in the PTA syntax. The default value is `3`.
- The maximum number of bind requests the PTA directory server can send simultaneously to the authenticating directory server within a single connection. In the PTA syntax, this parameter is represented as `maxops`. The default is value is `5`.

- The time limit you want the PTA directory server to wait for a response from the authenticating directory server. In the PTA syntax, this parameter is represented as `timeout`. The default value is `300` seconds (five minutes).
- The version of the LDAP protocol you want the PTA directory server to use to connect to the authenticating directory server. In the PTA syntax, this parameter is represented as `ldver`. The default is `LDAPv3`.
- The time limit in seconds within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating directory server. The server will not close the connection unless a bind request is initiated and the server determines the timeout has been exceeded. If you do not specify this option or if only one authenticating directory server is listed in the `authDS` parameter, no time limit will be enforced. If two or more hosts are listed, the default is `300` seconds (five minutes). In the PTA syntax, this parameter is represented as `connlifetime`.

NOTE Although these parameters are optional, if you specify one of them, you need to specify them all, even if you use the default values.

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: add
add: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://authDS/subtree
[maxconns,maxops,timeout,ldver,connlifetime]
```

   Make sure there is a space between the `subtree` parameter, and the optional parameters.

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

```
"ldap://dirserver.example.com/o=NetscapeRoot 3,5,300,3,300"
```

   In this example, each of the optional parameters is set to its default value.

2. Use the `ldapmodify` command to import the LDIF file into the directory.
3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

PTA Plug-in Syntax Examples

This section contains the following examples of PTA Plug-in syntax in the `dse.ldif` file:

- Specifying One Authenticating Directory Server and One Subtree
- Specifying Multiple Authenticating Directory Servers
- Specifying One Authenticating Directory Server and Multiple Subtrees
- Using Non-Default Parameter Values
- Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

Specifying One Authenticating Directory Server and One Subtree

This example configures the PTA Plug-in to accept all defaults for the optional variables. This configuration causes the PTA directory server to connect to the authenticating directory server for all bind requests to the `o=NetscapeRoot` subtree. The hostname of the authenticating Directory Server is `configdir.example.com`.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying Multiple Authenticating Directory Servers

If the connection between the PTA directory server and the authenticating directory server is broken or the connection cannot be opened, the PTA directory server sends the request to the next server specified, if any. You can specify as many authenticating directory servers as required.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0:
ldap://configdir.example.com/ou=NetscapeRoot
nsslapd-pluginarg1:
ldap://config2dir.example.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying One Authenticating Directory Server and Multiple Subtrees

The following example configures the PTA directory server to pass through bind requests for more than one subtree (using parameter defaults):

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
```

```
nsslapd-pluginarg0: ldap://configdir.example.com/ou=NetscapeRoot
nsslapd-pluginarg1:
ldap://configdir.example.com/dc=example,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

Using Non-Default Parameter Values

This example uses a non-default value (`10`) only for the maximum number of connections parameter `maxconns`. Each of the other parameters is set to its default value. However, because one parameter is specified, all parameters must be defined explicitly in the syntax.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/ou=NetscapeRoot
10,5,300,3,300
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

If you want to specify a different pass-through subtree and optional parameter values for each authenticating directory server, you must specify more than one LDAP URL/optional parameters pair. Separate the LDAP URL/optional parameter pairs with a single space as follows.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0:
ldap://configdir.example.com/ou=NetscapeRoot 7,7,300,3,300
nsslapd-pluginarg1:
ldap://config2dir.example.com/dc=example,dc=com 7,7,300,3,300
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: pass through authentication plugin
```

Chapter 17

Using the Attribute Uniqueness Plug-in

The Attribute Uniqueness Plug-in can be used to ensure that the attributes you specify always have unique values in the directory. You must create a new instance of the plug-in for every attribute for which you want to ensure unique values.

Red Hat Directory Server (Directory Server) provides a UID Uniqueness Plug-in that can be used to manage the uniqueness of the `uid` attribute.

This chapter describes the Attribute Uniqueness Plug-in and the UID Uniqueness Plug-in in the following sections:

- Overview of the Attribute Uniqueness Plug-in (page 531)
- Overview of the UID Uniqueness Plug-in (page 533)
- Attribute Uniqueness Plug-in Syntax (page 533)
- Creating an Instance of the Attribute Uniqueness Plug-in (page 536)
- Configuring Attribute Uniqueness Plug-ins (page 537)
- Attribute Uniqueness Plug-in Syntax Examples (page 541)
- Replication and the Attribute Uniqueness Plug-in (page 543)

Overview of the Attribute Uniqueness Plug-in

The Attribute Uniqueness Plug-in is a preoperation plug-in. This means that the plug-in checks all update operations before the server performs an LDAP operation. The plug-in determines whether the operation applies to an attribute and a suffix that you have configured it to monitor.

If an update operation applies to an attribute and suffix monitored by the plug-in and it would cause two entries to have the same attribute value, then the server terminates the operation and returns an `LDAP_CONSTRAINT_VIOLATION` error to the client.

The Attribute Uniqueness Plug-in performs a check on:

- A single attribute
- One or several subtrees

If you want to check uniqueness of several attributes, you must create a separate instance of the plug-in for each attribute you want to check.

You can also configure how the Attribute Uniqueness Plug-in operates:

- It can check every entry in the subtrees you specify.

  For example, if your company, `example.com`, hosts the directories for `example_a.com` and `example_b.com`, when you add an entry such as `uid=jdoe,ou=people,o=example_a,dc=example,dc=com`, you need to enforce uniqueness only in the `o=example_a,dc=example,dc=com` subtree. You can do this by listing the DN of the subtree explicitly in the UID Uniqueness Plug-in configuration.

  This configuration option is explained in more detail in "Specifying a Suffix or Subtree," on page 539.

- You can specify an object class pertaining to an entry in the DN of the updated entry and perform the uniqueness check on all the entries beneath it.

  This option is useful in hosted environments. For example, when you add an entry such as `uid=jdoe,ou=people,o=example_a,dc=example,dc=com`, you can enforce uniqueness under the `o=example_a,dc=example,dc=com` subtree without listing this subtree explicitly in the configuration but, instead, by indicating a *marker object class*. If you specify that the marker object class is `organization`, the uniqueness check algorithm locates the entry in the DN that has this object class (`o=example_a`) and performs the check on all entries beneath it.

  Additionally, you can specify to check uniqueness only if the updated entry includes a specified object class. For example, you could specify to perform the check only if the updated entry includes `objectclass=inetorgperson`.

  This configuration option is explained in more detail in "Using the markerObjectClass and requiredObjectClass Keywords," on page 540.

If you intend to use the Attribute Uniqueness Plug-in in a replicated environment, refer to "Replication and the Attribute Uniqueness Plug-in," on page 543.

Overview of the UID Uniqueness Plug-in

Directory Server provides an instance of the Attribute Uniqueness Plug-in, the UID Uniqueness Plug-in. By default, the plug-in ensures that values given to the `uid` attribute are unique in the suffix you configured when installing the directory (the suffix corresponding to the `userRoot` database).

You can change the configuration to specify additional suffixes or subtrees or by specifying to perform the check only under entries that contain a specified object class. The syntax and configuration of the UID Uniqueness Plug-in is the same as for any other attribute. For more information on the configuration changes you can make, see "Configuring Attribute Uniqueness Plug-ins," on page 537 .

By default, the Uid Uniqueness plug-in is disabled because it affects the operation of multi-master replication. For information on using the attribute uniqueness plug-in in a replicated environment, refer to "Replication and the Attribute Uniqueness Plug-in," on page 543.

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. The differences are highlighted in the sections below.

Use the following syntax to perform the uniqueness check under a suffix or subtree:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: descriptive_plugin_namee
nsslapd-pluginPath: /opt/redhat-ds/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute_name
nsslapd-pluginarg1: dn1
[ nsslapd-pluginarg2: dn2 ]
nsslapd-plugin-depends-on-type: database
```

```
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

NOTE

- You can specify any name you like in the `cn` attribute to name the plug-in. The name should be descriptive. This attribute does *not* contain the name of the attribute which is checked for uniqueness.
- You can specify only one attribute on which the uniqueness check will be performed.
- You can specify several DNs of suffixes or subtrees in which you want to perform the uniqueness check by incrementing the `nsslapd-pluginarg` attribute suffix by 1 each time.

The variable components of the Attribute Uniqueness Plug-in syntax are described in Table 17-1.

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
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: descriptive_plugin_name
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute=attribute_name
nsslapd-pluginarg1: markerObjectClass=objectclass1
[ nsslapd-pluginarg2: requiredObjectClass=objectclass2 ]
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

NOTE

- You can specify any name you like in the `cn` attribute to name the plug-in. The name should be descriptive. This attribute does *not* contain the name of the attribute which is checked for uniqueness.
- You can specify only one attribute on which the uniqueness check will be performed.
- If the `nsslapd-pluginarg0` attribute begins with `attribute=`*attribute_name*, then the server expects that the `nsslapd-pluginarg1` attribute will include a `markerObjectClass`.

The variable components of the attribute uniqueness plug-in syntax are described in Table 17-1.

Table 17-1 Attribute Uniqueness Plug-in Variables

Variable	Definition
descriptive_plugin_name	Specifies the name of this instance of the Attribute Uniqueness Plug-in. You do not have to include the name of the attribute for which you want to ensure uniqueness, but it is advisable. For example, `cn=mail uniqueness,cn=plugins,cn=config`.
extension	File extension for the plug-in. The extension is always `.sl` on HP-UX and `.so` on all other UNIX platforms.
state	Defines whether the plug-in is enabled or disabled. Acceptable values are `on` or `off` . See "Turning the Plug-in On or Off," on page 539, for more information.
attribute_name	The name of the attribute for which you want to ensure unique values. You can specify one attribute name only.
dn	The DN of the suffix or subtree in which you want to ensure attribute uniqueness. You can specify several suffixes or subtrees by incrementing the suffix of the `nsslapd-pluginarg` attribute by 1 for each additional suffix or subtree.
`attribute=`*attribute_name*	The name of the attribute for which you want to ensure unique values. You can specify one attribute name only.
`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.

Table 17-1 Attribute Uniqueness Plug-in Variables *(Continued)*

Variable	Definition
`requiredObjectClass=`*objectclass2*	*Optional.* When you use the `markerObjectClass` keyword to specify the scope of the uniqueness check instead of a DN, you can optionally 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.

Creating an Instance of the Attribute Uniqueness Plug-in

If you want to ensure that a particular attribute in your directory always has unique values, you must create an instance of the Attribute Uniqueness Plug-in for the attribute you want to check. For example, if you want to ensure that every entry in your directory that includes a `mail` attribute has a unique value for that attribute, you must create a mail uniqueness plug-in.

To create an instance of the Attribute Uniqueness Plug-in, you must modify the `dse.ldif` file 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 "Attribute Uniqueness Plug-in Syntax," on page 533.

For example, to create an instance the Attribute Uniqueness Plug-in for the mail attribute, you would perform the following steps:

1. In the `dse.ldif` file, locate the entry for the UID Uniqueness Plug-in, `cn=uid uniqueness,cn=plugins,cn=config`.
2. Add the following lines for the mail uniqueness plug-in entry before or after the UID Uniqueness Plug-in entry:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath:
/opt/redhat-ds/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
```

```
nsslapd-pluginarg1: dc=example,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

3. Restart Directory Server.

In this example, the uniqueness check will be performed on every entry in the `dc=example,dc=com` entry that includes the `mail` attribute.

Configuring Attribute Uniqueness Plug-ins

This section explains how to use Directory Server Console to view the plug-ins configured for your directory and how to modify the configuration of the Attribute Uniqueness Plug-ins.

Viewing Plug-in Configuration Information

From the Directory Server Console, you can display the configuration entry for Attribute Uniqueness Plug-ins as follows:

1. In the Directory Server Console, click the Directory tab.
2. In the left navigation tree, expand the config folder, then the plugins folder.

   The list of plug-ins is displayed in the right navigation window. You should see the UID Uniqueness Plug-in and any other Attribute Uniqueness Plug-ins that you created following the example given in "Creating an Instance of the Attribute Uniqueness Plug-in," on page 536.
3. In the right navigation window, double-click the plug-in entry you want to look at.

   The Property Editor is displayed. It contains a list of all the attributes and values for the plug-in.

Configuring Attribute Uniqueness Plug-ins from the Directory Server Console

You can update plug-in configuration from Directory Server Console in several ways:

- From the Property Editor.

  Display the Property Editor, as explained in "Viewing Plug-in Configuration Information," on page 537, and edit the attribute value fields.

- From the Configuration tab.

To modify an Attribute Uniqueness Plug-in configuration from the Directory Server Console Configuration tab:

1. In the Directory Server Console, select the Configuration tab; then, in the navigation tree, expand the Plugins folder, and select the Attribute Uniqueness Plug-in that you want to modify.

   The configuration parameters for the plug-in are displayed in the right pane.

2. To turn the plug-in on or off, check or clear the Enable Plugin checkbox.

3. To add a suffix or subtree, click Add, and type a DN in the blank text field.

   If you do not want to add a DN, you can use the `markerObjectClass` keyword. If you use this syntax, you can click Add again to specify a requiredObjectClass as described in "Attribute Uniqueness Plug-in Syntax," on page 533.

NOTE You must not add an attribute name to the list. If you want to check the uniqueness of other attributes, you must create a new instance of the Attribute Uniqueness Plug-in for the attribute you want to check. For information, refer to "Creating an Instance of the Attribute Uniqueness Plug-in," on page 536.

4. To delete an item from the list, place the cursor in the text field that you want to delete, and click Delete.

5. Click Save to save your changes.

Configuring Attribute Uniqueness Plug-ins from the Command-Line

This section provides information about configuring the plug-in from the command-line. It covers the following tasks:

- Turning the Plug-in On or Off
- Specifying a Suffix or Subtree
- Using the markerObjectClass and requiredObjectClass Keywords

Turning the Plug-in On or Off

To turn the plug-in on from the command-line, you must create an LDIF file that contains the following LDIF update statements:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

Use the `ldapmodify` command to import the LDIF file into the directory. For detailed information on the `ldapmodify` command, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

To disable the plug-in, change the LDIF update statements to replace the `nsslapd-pluginenabled: on` statement with the `nsslapd-pluginenabled: off` statement.

Whenever you enable or disable the plug-in, you must restart the server. For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

Specifying a Suffix or Subtree

You specify the suffix or subtrees under which you want the plug-in to ensure attribute uniqueness by using the `nsslapd-pluginarg` attribute in the entry defining the plug-in.

You can specify the subtree or subtrees by creating and LDIF file that contains update statements similar to those shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
changetype: add
nsslapd-pluginarg2: ou=Engineering,dc=example,dc=com
nsslapd-pluginarg3: ou=Sales,dc=example,dc=com
```

This example LDIF file will check 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`.

Use the `ldapmodify` command to import the LDIF file into the directory. For detailed information on the `ldapmodify` command, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

Whenever you make this type of configuration change, you must restart the server. For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 37.

Using the markerObjectClass and requiredObjectClass Keywords

Instead of specifying a suffix or subtree in the configuration of an Attribute Uniqueness Plug-in, you can specify to perform the check under the entry belonging to the DN of the updated entry that has the object class specified 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, you can create an LDIF file such as the one shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /opt/redhat-ds/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=ou
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

If you do not want the server to check every entry under the organizational unit entry, you can limit the scope by specifying to perform the check only if the updated entry contains a specified object class.

For example, if you check the uniqueness of the `mail` attribute, it is probably necessary to perform the check only when you add or modify entries that contain the `person` or `inetorgperson` object class.

You can 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
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /opt/redhat-ds/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=ou
nsslapd-pluginarg2: requiredObjectClass=person
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

You cannot repeat the `markerObjectClass` or `requiredObjectClass` keywords by incrementing the counter in the `nsslapd-pluginarg` attribute suffix.

NOTE The `nsslapd-pluginarg0` attribute always contains the name of the attribute for which you want to ensure uniqueness.

Attribute Uniqueness Plug-in Syntax Examples

This section contains examples of Attribute Uniqueness Plug-in syntax in the `dse.ldif` file.

- Specifying One Attribute and One Subtree
- Specifying One Attribute and Multiple Subtrees

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
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /opt/redhat-ds/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=example,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

Specifying One Attribute and Multiple Subtrees

This example configures the plug-in to ensure the uniqueness of the `mail` attribute under the `l=Chicago,dc=example,dc=com` and `l=Boston,dc=example,dc=com` subtrees.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /opt/redhat-ds/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: l=Chicago,dc=example,dc=com
nsslapd-pluginarg2: l=Boston,dc=example,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 7.1
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: Enforce unique attribute values.
```

NOTE The nsslapd-pluginarg0 attribute always contains the name of the attribute for which you want to ensure uniqueness. All other occurrences of the nsslapd-pluginarg (nsslapd-pluginarg1 to nsslapd-pluginargx) 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 would be allowed:

```
mail=bjensen,l=Chicago,dc=example,dc=com
mail=bjensen,l=Boston,dc=example,dc=com
```

If you want to ensure that only one instance of a value exists under both subtrees, you need to configure the plug-in to ensure uniqueness for the entire dc=example,dc=com subtree.

Replication and the Attribute Uniqueness Plug-in

When you use the Attribute Uniqueness Plug-ins on Directory Servers involved in a replication agreement, you must think carefully about how to configure the plug-in on each server.

Consider the following cases:

- Simple replication with one supplier and one or several consumers.
- Complex replication with multiple masters.

Attribute Uniqueness Plug-ins do not perform any checking on attribute values when an update is performed as part of a replication operation.

Simple Replication Scenario

Because all modifications by client applications are performed on the supplier server, the Attribute Uniqueness Plug-in should be enabled on the supplier. It is unnecessary to enable it on the consumer server.

Enabling the Attribute Uniqueness Plug-in on the consumer will not prevent Directory Server from operating correctly but is likely to cause a performance degradation.

Multi-Master Replication Scenario

In a multi-master replication scenario, the masters act both as suppliers and consumers of the same replica. Because multi-master replication uses a loosely consistent replication model, enabling an Attribute Uniqueness Plug-in on one of the servers is not sufficient to ensure that attribute values will be unique across both masters at any given time. Therefore, enabling an Attribute Uniqueness Plug-in on one server can cause inconsistencies in the data held on each replica.

However, you can use an Attribute Uniqueness Plug-in, providing both of the following conditions are met:

- The attribute on which you are performing the uniqueness check is a naming attribute.
- The Attribute Uniqueness Plug-in is enabled on both masters.

When these conditions are met, attribute uniqueness conflicts are reported as naming conflicts at replication time. Naming conflicts require manual resolution. For information on how to resolve replication conflicts, refer to "Solving Common Replication Conflicts," on page 370.

Chapter 18

Windows Sync

The Windows Sync feature allows synchronization of adds, deletes and changes in groups, user entries, and their passwords between Red Hat Directory Server and both Microsoft Active Directory and Microsoft Windows NT 4.0 Server. It provides an efficient and effective way to maintain consistent directory information across the enterprise.

This chapter covers how to configure and use Windows Sync in the following sections:

- About Windows Sync (page 545)
- Installing Sync Services (page 551)
- Uninstalling the Sync Services (page 558)
- Using Windows Sync (page 562)
- Troubleshooting (page 570)

About Windows Sync

The complete Windows Sync feature is implemented in three parts:

- **Directory Server Windows Sync code.** The server itself contains code that performs a large proportion of the sync functionality. This code is closely integrated with the server's own native multi-master replication plug-in. The same changelog that is used for multi-master replication is also used to replay outbound changes that pertain to synchronized entries. The corresponding changes are made in the Windows Server via LDAP. The server also performs LDAP search operations against its Windows Server peer in order to synchronize inbound changes made to Windows user entries.

- **Password Sync Service.** This is an application that must be installed on the Active Directory machine (or primary domain controller in the case of NT4). Its purpose is to capture password changes for Windows users and relay those changes back to the Directory Server via LDAP with SSL, for security.
- **NT4 LDAP Service.** This is a special LDAP server application that must be installed on the primary domain controller for NT4 sync. It is only used for NT4 and is not needed for Active Directory deployments. The purpose of the NT4 LDAP Service is to provide a similar view of users and groups as is available via LDAP from Active Directory. This allows almost all of the Directory Server Windows Sync code to be the same for both Active Directory and NT4.

NOTE The term *Windows Server* refers to both Active Directory and NT4 (which assumes that the NT4 LDAP Service is installed). Only where configuration or behavior is different between Active Directory and NT4 Server will the type of Windows server be specified.

Figure 18-1 and Figure 18-2 show the relationship between Red Hat Directory Server and Active Directory and Windows NT4 Server primary domain controller (PDC), respectively.

Figure 18-1 Active Directory - Directory Server Synchronization Process

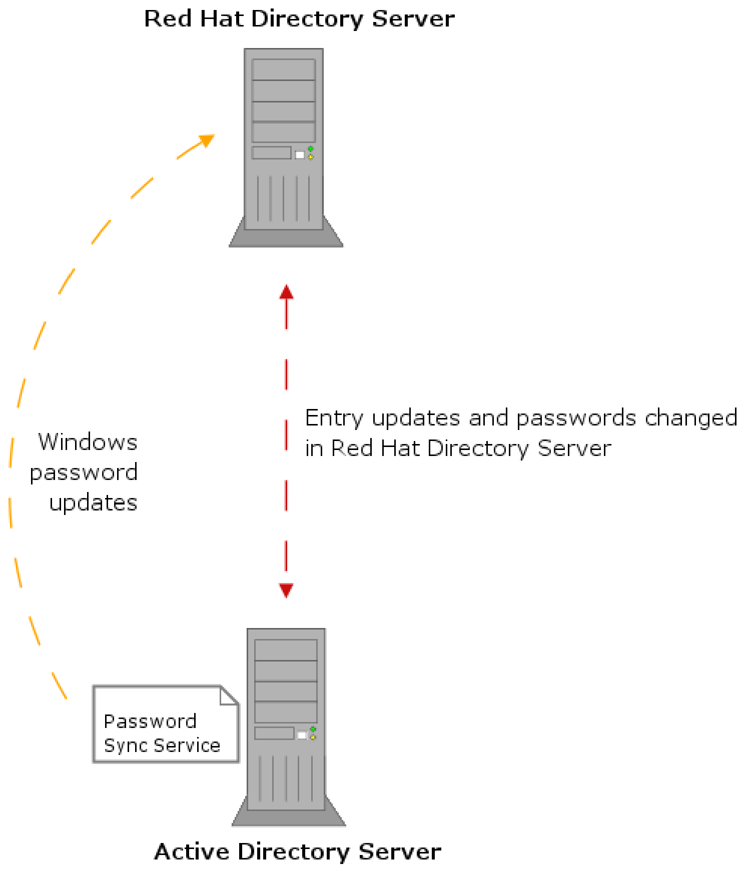
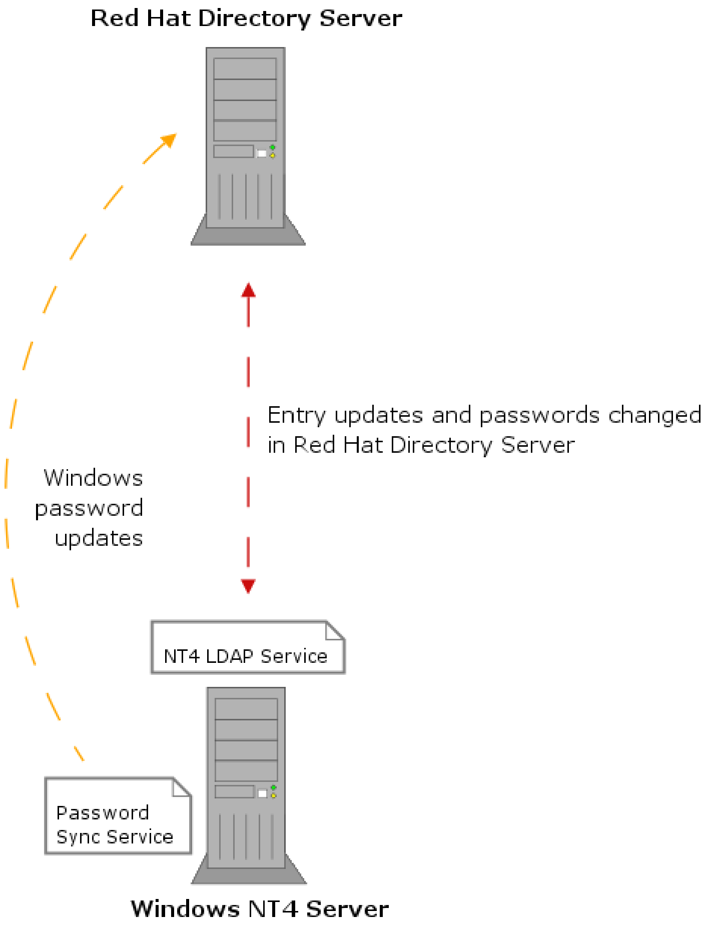
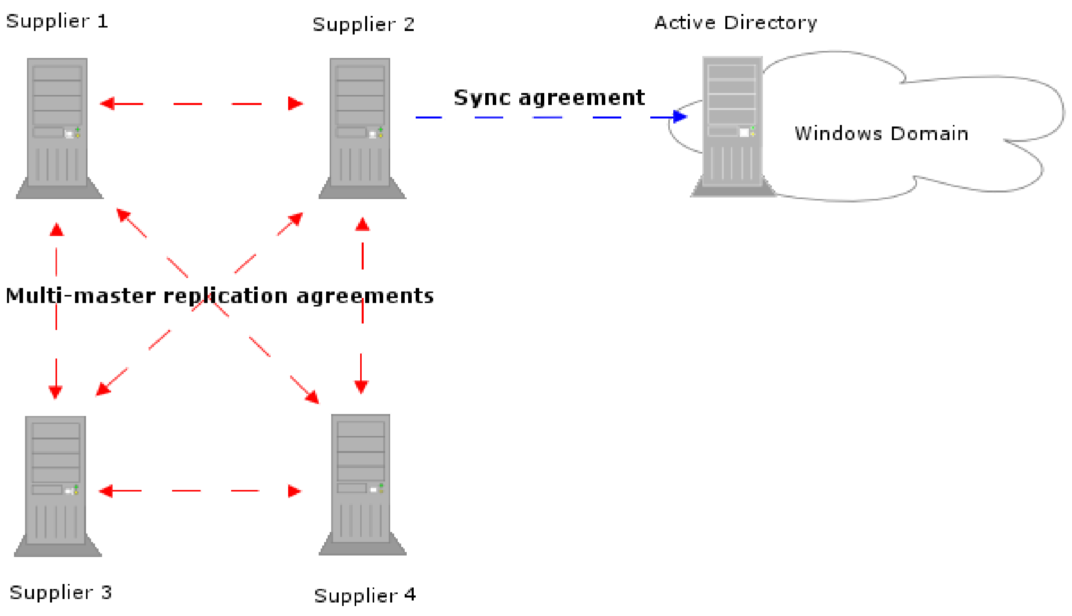


Figure 18-2 Windows NT4 Server - Directory Server Synchronization Process



Windows Sync is compatible with Directory Server's multi-master replication facilities. Figure 18-3 shows this arrangement:

Figure 18-3 Multi-Master Directory Server - Windows Domain Synchronization



How Windows Sync Works

Synchronization is configured and controlled by means of one or more *synchronization agreements*. These are similar in purpose to replication agreements and contain a similar set of information, including the host name and port number for the Windows Server.

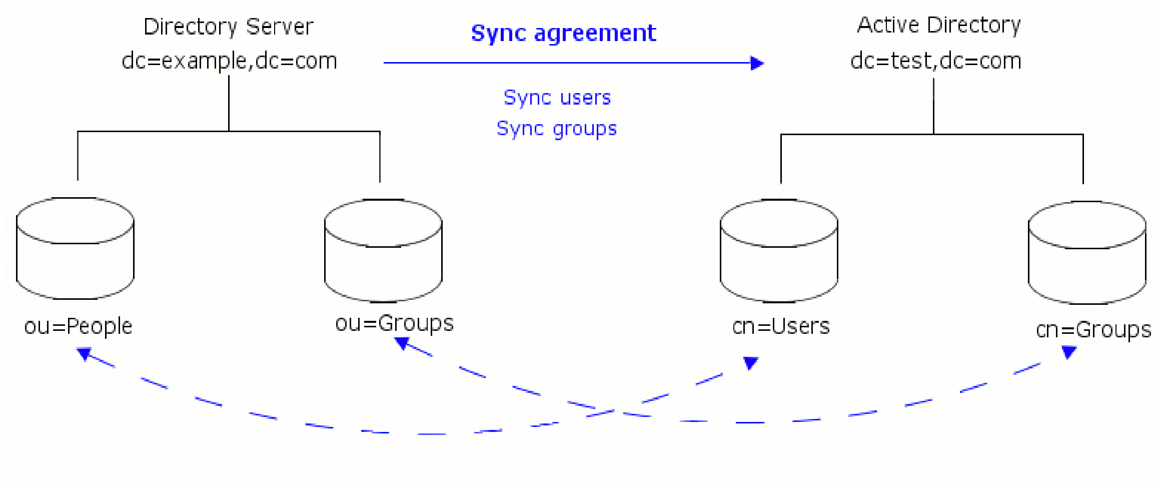
The Directory Server connects to its peer Windows Server via LDAP and SSL to both send and retrieve updates.

The unit of synchronization is a subtree. A single Windows subtree can be synchronized to a single Directory Server subtree and vice versa. The Windows and Directory Server subtree DNs are specified in the sync agreement. All entries within the respective subtrees are candidates for synchronization, including entries that are not immediate children of the subtree root. It's important to note however that any descendent container entries will need to be created separately by the administrator. Windows Sync never creates container entries itself.

Windows Sync provides some control over which entries are synchronized. This is intended to allow administrators to determine that only a subset of all the entries should be subject to synchronization and to give sufficient flexibility to support different deployment scenarios. Therefore, certain rules are applied to determine first if a particular entry should be synchronized and second if a given new entry should be created in the peer server.

Specifically, only entries with user or group object classes are synchronized from Windows to Directory Server. In addition, two flags with agreement scope allow the creation of new entries for newly found entries in the Windows Server to be enabled or disabled. One flag controls new entry creation for users while the other controls new entry creation for groups. Similarly, only entries with the necessary object classes and attribute values in the Directory Server will be synchronized.

The Directory Server peer maintains a *replication changelog*, a database that records the modifications that have occurred. The changelog is used by Windows Sync to generate outbound changes made to the Windows Server. Therefore, a changelog and the associated replica object must be configured in a Directory Server before it may run Windows Sync.



During normal operation, all the updates made to entries in the Directory Server that need to be sent to the Windows Server are generated via the changelog. However, when the server is initially configured or after major changes to its content, it is necessary to initiate a *re-synchronization process*. For re-synchronization, the entire contents of synchronized subtree in the Directory Server is examined and, if necessary, sent to the Windows Server. This is done without using the changelog.

Inbound changes, that is changes to entries in the Windows Server, are found by using Active Directory's 'Dirsync' search feature. Because there is no changelog to use, it is necessary to issue the Dirsync search periodically. The default interval is five minutes. The administrator may also trigger an immediate Dirsync search by right-clicking on the sync agreement and selecting "Send and Receive Updates Now". Use of the Dirsync search ensures that only those entries that have changed since the previous search are retrieved. However, in the case of a server or where there have been major changes to its content, a Dirsync search that returns all entries (and not just recently changed entries) can be performed. This full Dirsync is done whenever the administrator initiates the 're-synchronization' process mentioned above. In some situations, it may be unappealing to wait up to five minutes for the next Dirsync search to occur. During this time, recent changes made on the Windows Server will not be brought over to the Directory Server. In this case, the administrator can manually initiate an immediate Dirsync search.

In addition to the entry synchronization mechanisms discussed above, the Password Sync Service is needed to catch password changes made on the Windows server. Without the Password Sync Service, it would be impossible to have inbound password sync because passwords are hashed once stored in Active Directory, and the hashing function is incompatible with that used by Directory Server. Also, it is not possible to retrieve password values from a Windows Server externally. Outbound passwords are synchronized along with other entry attributes using a special Directory Server facility for retaining the plaintext password values in the changelog.

Installing Sync Services

There are two services that can be installed on your Windows machine to synchronize more aspects of your Directory Server with a Windows Server.

Password Sync must be installed on the Windows Server. It synchronizes password changes made on the Windows Server with the corresponding entries' passwords on the Directory Server.

The NT4 LDAP Service is installed only on Windows NT4 Server to allow synchronization operations over LDAP between the Directory Server and Windows NT4 Server.

Installing and Configuring the Password Sync Service

NOTE For Windows NT4 servers, Windows Sync must be configured to a PDC, and all sync services must be installed on a primary domain controller (PDC). Synchronization will not function properly on a non-PDC machine.

NOTE On Windows 2000, password complexity policies and disabled by default. They must be enabled in order for the password hook DLL to be triggered. Refer to the appropriate Windows documentation for more information.

1. Copy the `PassSync.msi` file that contains the Password Sync utility to the Windows machine.
2. Double-click on the `PassSync.msi` file to install it.

   The Password Sync Setup window will appear. Hit "Next" to begin installing.
3. Fill in the Directory Server hostname, secure port number, user name (such as `cn=sync manager,cn=config`), the certificate token (password), and the search base (e.g., `ou=People,dc=example,dc=com`).

NOTE This user must have write access to the `userpassword` attribute for all users in the synched Directory Server subtree.

Hit “Next,” then “Finish” to install Password Sync.

4. Reboot the Windows machine to start Password Sync.

NOTE	You must reboot the Windows machine. Without rebooting, the password hook DLL will not be enabled, and password synchronization will not function.

Password Sync is installed in `C:\Program Files\Red Hat Directory Password Synchronization`, and the `passsync.exe` is the only file in the installation directory.

The following `.dll`s are installed in `C:\winnt\system32` and utilized by Password Sync:

`passhook.dll`	`nsldap32v50.dll`
`nsldapssl32v50.dll`	`libplc4.dll`
`nsldappr32v50.dll`	`nss3.dll`
`libnspr4.dll`	`ssl3.dll`

```
libplds4.dll          softokn3.dll
```

The Password Sync Service runs as a Windows service, which means that it can be started, stopped, and controlled by the `net start|stop` command, the Services Control Panel applet, and other Windows Services management mechanisms.

Changed passwords are captured even if the Password Sync Service is not running. If the Password Sync Service is restarted, the password changes are sent to Directory Server immediately.

Reconfiguring the Password Sync Service

To reconfigure Password Sync, open the Windows Services panel, highlight Password Sync, and select Modify. This will lead you back through the configuration screens.

Setting Up SSL for the Password Sync Service

Next, set up certificates that Password Sync Service will use SSL to access the Directory Server:

1. Download `certutil.exe` if you do not already have it installed on your machine. It is available from `ftp://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/`. See "Managing SSL and SASL," on page 425, for more information on SSL.
2. Create a new `cert8.db` and `key.db` using `certutil.exe` on the Password Sync machine.

   ```
   certutil.exe -d . -N
   ```
3. On your Directory Server, export the server certificate using `pk12util`.

   ```
   pk12util -d . -P slapd-serverID -o servercert.pfx -n
   Server-Cert
   ```
4. Copy the exported certificate from the Directory Server to the Windows machine.
5. Import the server certificate from the Directory Server into the new certificate database using `pk12util.exe`.

   ```
   pk12util.exe -d "C:\Program Files\Red Hat Directory Password
   Synchronization" -i servercert.pfx
   ```
6. Give "trusted peer" status to the server.

```
certutil.exe -d "C:\Program Files\Red Hat Directory Password
Synchronization" -M -n Server-Cert -t "P,P,P"
```

NOTE If any Windows user accounts exist when you first install Password Sync, then the passwords for those user accounts cannot be synchronized until they are changed because Password Sync cannot de-encrypt a password once it has been encrypted.

Installing and Configuring the NT4 LDAP Service

For Windows NT4 servers, Windows Sync must be configured to a PDC, and all sync services must be installed on a PDC. Synchronization will not function properly on a non-PDC machine.

1. Double click the `ntds.msi` file to install.

   This is downloaded into `C:\Program Files\Red Hat Directory Synchronization`.

2. Open `C:\Program Files\Red Hat Directory Synchronization\bin`, and double-click on `installuseresync.bat`. This will set up the LDAP Service as a Windows service.

3. Open `C:\Program Files\Red Hat Directory Synchronization\conf`. Modify the `usersync.conf` file to reflect the Directory Server port, SSL port, and hostname.

   The only required parameters are `server.net.admin.password` and `server.db.partition.suffix.usersync`. In the following code example, these parameters are in bold.

   `server.net.admin.password` sets the password of the account `uid=admin,ou=system`. This is the bind ID used by the Directory Server to send updates to the NT4 Server.

   `server.db.partition.suffix.usersync` sets the suffix for the NT4 Server, since NT4 Server does not use the same directory tree structure used by Directory Server. This can be set to the same suffix as the Directory Server to which you are synchronizing.

NOTE After the service is installed and started the first time the password can only be changed via an LDAP modify operation, not the configuration file.

```
#server.net.ldap.port=389
#server.net.ldaps.port=636
server.net.admin.password=password33
javax.net.ssl.keyStore=c:\\keystore
javax.net.ssl.keyStorePassword=password
server.net.ldaps.enable=true
server.db.partition.suffix.usersync=dc=example,dc=com
# do not modify beyond this point
server.schemas =
org.apache.ldap.server.schema.bootstrap.CoreSchema
org.apache.ldap.server.schema.bootstrap.CosineSchema
org.apache.ldap.server.schema.bootstrap.ApacheSchema
org.apache.ldap.server.schema.bootstrap.InetorgpersonSchema
org.apache.ldap.server.schema.bootstrap.JavaSchema
org.apache.ldap.server.schema.bootstrap.SystemSchema
org.apache.ldap.server.schema.bootstrap.UsersyncSchema
server.db.partitions=usersync
server.db.partition.class.usersync=org.apache.ldap.server.NetAPI
Partition
server.db.partition.indices.usersync=ou objectClass
server.db.partition.attributes.usersync.ou=usersync
server.db.partition.attributes.usersync.objectClass=top
organizationalUnit extensibleObject
```

It is not necessary to set the port number. The defaults for both the regular and secure ports are used automatically. Only one port can be used by the LDAP Service; therefore, if you specify a port, you must specifiy either the regular port or the secure port, not both.

If you use SSL, you must signal the service to use SSL by setting the following parameter and provide information for the keystore:

```
server.net.ldaps.enable=true
javax.net.ssl.keyStore=c:\\keystore
javax.net.ssl.keyStorePassword=password
```

The keystore information is set in the following step.

4. To enable SSL, do the following.

a. Create a self-signed certificate using Java `keytool`:

```
C:\>keytool -genkey -alias ldap -keyalg RSA -validity 3650
-keystore c:\keystore

Enter keystore password:  password
What is your first and last name?
[Unknown]:  directory.example.com
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:  example.com
What is the name of your City or Locality?
[Unknown]:  Boston
What is the name of your State or Province?
[Unknown]:  MA
What is the two-letter country code for this unit?
[Unknown]:  US
Is CN=directory.example.com, OU=Unknown, O=example.com,
L=Boston, ST=MA, C=US correct?
[no]:  yes

Enter key password for <ldap>
(RETURN if same as keystore password):
```

Use the same password for the certificate and keystore.

The first and last name field should be the fully qualified domain name of the machine running the NT4 LDAP Service. If a different value is entered as a security precaution, you must disable the "check hostname against name in certificate" option in your Directory Server SSL configuration.

b. Export the CA certificate you created so that it can be imported into Directory Server.

```
c:\>keytool -export -alias ldap -keystore c:\keystore -rfc
-file c:\ca.cer

Enter keystore password:  password
Certificate stored in file <ca.cer>
```

c. Copy this file, `ca.cer`, to your Directory Server machine.

d. Import the CA using the Console.

In the Tasks tab, select Manage Certificates. Open the CA Certs tab, hit the "Install" button, and import the CA certificate from the directory where you copied it.

5. Open the Services control panel, and right-click on User Sync Service. Select start.

The NT4 LDAP Service runs as a Windows Service, which means that it can be started, stopped, and controlled by the `net start|stop` command, the Services Control Panel applet, and other Windows Services management mechanisms.

Entry changes are not "stored" by the NT4 LDAP Service when the service is stopped, but the current state of the entries is synchronized automatically when the LDAP Service is restarted.

Uninstalling the Sync Services

To uninstall the Password Sync Service:

1. Open the Add/Remove Programs utility.
2. Select click remove to uninstall the Password Sync Service.
3. If you configured SSL for the Password Sync, then the `cert8.db` and `key3.db` databases that were created were not removed when you uninstall Password Sync. Delete these files by hand.

To uninstall the NT4 LDAP Service:

1. Open `C:\Program Files\Red Hat Directory Synchronization\bin`, and double-click on `uninstalluseresync.bat`.
2. Open the Add/Remove Programs utility.
3. Select click remove to uninstall the User Sync Service.

Configuring Windows Sync

Step 1: Configure SSL on the Directory Server

To configure the Directory Server to run in SSL. You can use the `certutil` utility to create self-signed certificates or obtain and install certificates to enable SSL; for more information, see chapter 11, "Managing SSL and SASL."

You should have the following things created and installed on both your Directory Server and your Windows sync peers:

- CA certificate, shared between the Directory Server and Windows server
- Directory Server certificate that is accessible by your sync services

Step 2: Configure SSL on Active Directory (Active Directory only).

To configure SSL on Active Directory, see the appropriate user documentation. It is not necessary to configure SSL for NT4 Server; SSL is enabled when configuring the NT4 LDAP Service.

Step 3: Install and Configure the Password Sync Service

Password Sync can be installed on any Windows machine to synchronize Windows passwords. Passwords can only be synchronized if both your Directory Server and Windows Server are running in SSL, the sync agreement is configured over an SSL connection, and you have configured certificate databases for Password Sync to access. See "Installing and Configuring the Password Sync Service," on page 552, for information on installing and configuring Password Sync.

Step 4: Configure the NT4 LDAP Service (Windows NT4 Server Only)

Install the LDAP Service on the Windows NT4 Server, set it up as a Windows service, and modify the configuration file for your Directory Server information. See "Installing and Configuring the NT4 LDAP Service," on page 555, for more information.

Step 5: Select or Create the Sync Identity

The Windows user specified in the sync agreement, which the Directory Server will use to bind for sync operations, should be a member of the Domain Admins group (or have equivalent privileges). A member of this group will have full privileges within the domain, but will not necessarily have privileges within other domains in the Active Directory forest, enhancing security. This limits the extent of the Windows directory that can be affected by the sync ID to only the synchronized subtree.

TIP	It may be useful to lock this admin user from being able to logon to the domain from a physical location. The entry would be able to modify the directory entries, but no one could use that entry to perform a console logon to any machine in the domain. Refer the Windows documentation for more information.

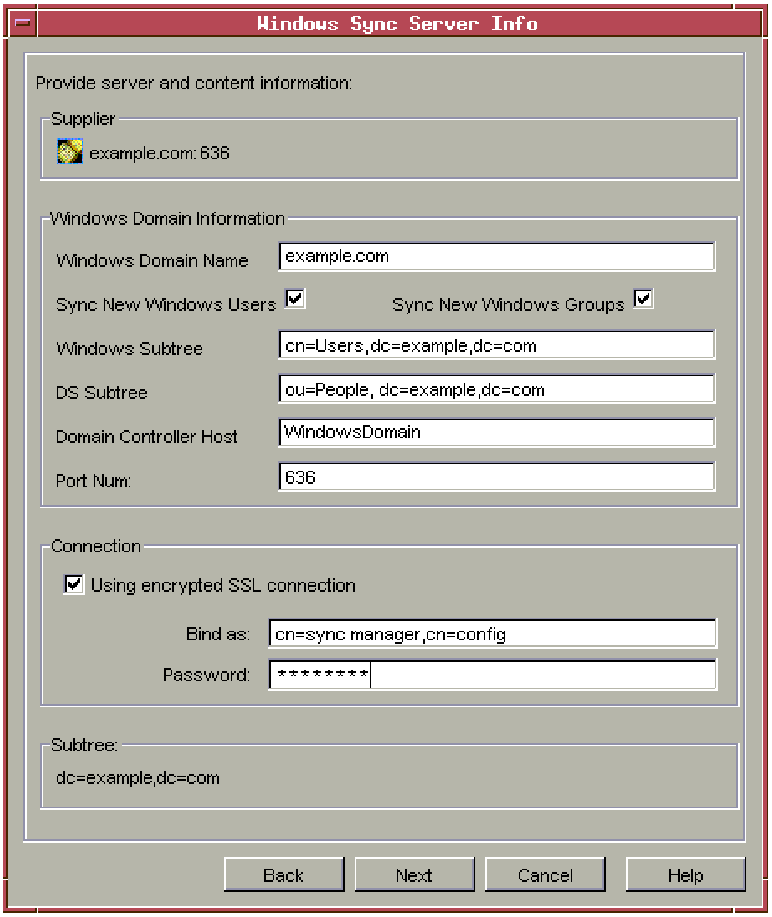
The user specified in the Password Sync and NT4 LDAP Services should be a a special user that has write access to entries and passwords but, for security reasons, should not be Directory Manager. Also, this user should not be under the synchronized subtree. For information on creating a special sync ID, see "Creating the Supplier Bind DN Entry," on page 318.

Step 6: Create the Synchronization Agreement

To create a synchronization agreement:

1. In the Directory Server Console, select the Configuration tab.
2. In the left-hand navigation tree, right-click on the suffix to sync, and select New Synchronization Agreement. You can also highlight the suffix, and select Menu>Object>New Synchronization Agreement.

   This will start the Synchronization Agreement Wizard.
3. In the two fields, supply a name and description of your synchronization agreement. Hit “Next.”
4. The second screen reads “Windows Sync Server Info.” By default, your Directory Server hostname and port are visible at the top, under Supplier. At the very bottom of the screen, the name of the synched suffix, such as `dc=example,dc=com`, is displayed.



Provide server and content information:

Supplier

example.com:636

Windows Domain Information

Windows Domain Name	example.com
Sync New Windows Users	✓
Sync New Windows Groups	✓
Windows Subtree	cn=Users,dc=example,dc=com
DS Subtree	ou=People, dc=example,dc=com
Domain Controller Host	WindowsDomain
Port Num:	636

Connection

✓ Using encrypted SSL connection

Bind as:	cn=sync manager,cn=config
Password:	********

Subtree:

dc=example,dc=com

5. In the middle of the screen are fields for your Windows domain information. Fill in the domain name and the domain controller.

6. Select the checkbox(es) for the Windows entries you are going to synchronize.

- **Sync New Windows Users.** When enabled, all user entries found in Windows that are subject to the agreement will automatically be created in the Directory Server.
- **Sync New Windows Groups.** When enabled, all group entries found in Windows that are subject to the agreement will automatically be created in the Directory Server.

7. The Windows and Directory Server subtree information is automatically filled in; use the defaults to sync only users or change these as appropriate to sync groups or groups and users.
8. Check the "Using encrypted SSL connection" checkbox. The use of SSL is recommended for security reasons. You *must* use SSL if you are going to synchronize passwords because Active Directory will refuse to modify passwords unless the connection is SSL protected.
9. Fill in the authentication information in the "Bind as..." and "Password" fields with the sync manager information.
10. The last screen is a summary of your synchronization agreement. If you decide to change anything at this point, you can use the back buttons to get to the appropriate screen. If you are satisfied with your agreement, click "Done."

When you have finished, an icon representing the synchronization agreement is displayed under the suffix. This icon indicates that your synchronization agreement is set up.

Step 6: Begin Synchronization

After the sync agreement is created, you need to begin the synchronizaiton process. Select the sync agreement, right-click or open the Object menu, and select "Initiate re-synchronization." This will begin the synchronization process.

If synchronization stops for any reason, you can initiate re-synchronization by selecting this from the sync agreement menu.

Using Windows Sync

After your Windows Sync service has been installed and configured, you can begin using it to synchronize your user and group entries on your Directory Server and Windows NT4 Server or Active Directory servers.

This section deals with how to move existing entries between servers, how to create and delete entries, and checking the status of synchronization.

- Synchronized Entries
- Groups
- Manually Initiating Synchronization
- Checking Synchronization Status
- Modifying the Synchronization Agreement

- The Connection tab will let you change the bind DN and bind credentials for the sync manager. It will also show whether this is over an SSL connection. Finally, it shows whether new user and group entries will be created in the Directory Server.
- NT4-Specific Limitiations

Synchronized Entries

Special schema is applied to entries in the Directory Server that are subject to synchronization. This schema is very similar, but not identical, to that used by the old Netscape Directory Server 4.x NT Sync feature. Full schema details are available in *Red Hat Directory Server Schema Reference*.

First, there are provisions for identifying the corresponding entry for a given Windows entry. This is done with the `ntUniqueId` attribute, which contains the value of the `objectGUID` attribute for the corresponding entry. This attribute is totally under the control of the sync code and should not be modified manually.

Next, the `ntDomainUser` attribute holds the value of the `samAccountName` attribute from the corresponding Windows entry. (In the case of NT4, it matches the user name).

Finally, `ntUserCreateNewAccount` and `ntUserDeleteAccount` attributes control the life cycle of the corresponding Windows entry. Only if `ntUserCreateNewAccount` has the value `true` will a new entry be created in the Windows Server. Similarly, only if `ntUserDeleteAccount` has the value true will the corresponding entry be deleted when the Directory Server entry is deleted. These attributes allow the adminstrator to exercise fine-grain control over the life cycle of synchronized entries.

Table 18-1 shows the attributes that are mapped between the Directory Server and Windows Servers, and Table 18-2 shows the attributes that are the same between the Directory Server and Windows Servers.

Table 18-1 User Entry Schema Mapping between Directory Server and Windows Servers

Directory Server	Active Directory	Windows NT4 Server
cn	**name**	**usri_full_name**
ntUserDomainId	**sAMAccountName**	
ntUserHomeDir	homeDirectory	usri_home_dir
ntUserScriptPath	scriptPath	usri_script_path

Table 18-1 User Entry Schema Mapping between Directory Server and Windows Servers

Directory Server	Active Directory	Windows NT4 Server
ntUserLastLogon	lastLogon	usri_last_logon
ntUserLastLogoff	lastLogoff	usri_last_logoff
ntUserAcctExpires	accountExpires	usri_acct_expires
ntUserCodePage	codePage	usri_code_page
ntUserLogonHours	logonHours	usri_logon_hours
ntUserMaxStorage	maxStorage	usri_max_storage
ntUserProfile	profilePath	usri_profile
ntUserParms	userParameters	usri_parms
ntUserWorkstations	userWorkstations	usri_workstations

Table 18-2 User Entry Schema That Is the Same in Directory Server and Windows Servers

description	postOfficeBox
destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenName	registeredAddress
homePhone	sn
homePostalAddress	st
initials	street
l	telephoneNumber
mail	teletexTerminalIdentifier
mobile	telexNumber
o	title
ou	userCertificate
pager	x121Address
physicalDeliveryOfficeName	

When you create a Directory Server user from the Console (see "Creating Directory Entries," on page 49), there is an NT User tab in the New User dialog. Fill in this information to supply Windows attributes automatically.

You can add additional `ntUser` attributes either by using the Advanced button in the Console or by using `ldapmodify`; see "Modifying Entries Using ldapmodify," on page 62.

Groups

Similar to user entires, group entries are synchronized if they have the `ntGroup` and `mailgroup` object classes. There are also two attributes that control creation and deletion of group entries in the Windows Server: `ntGroupCreateNewAccount and ntGroupDeleteAccount.`

Group entries that are within the scope of the sync agreement will be synchronized in much the same way as user entries. In addition, the membership of groups is synchronized with the constraint that only those members that are also within the scope of the agreement are propagated. The result is that a group may contain members that are both within and without the scope of the agreement, but only the subset of members that are themselves within agreement scope are synchronized. The remaining members are left unchanged on both sides.

Table 18-3 shows the attributes that are mapped between the Directory Server and Windows Servers, and Table 18-4 shows the attributes that are the same between the Directory Server and Windows Servers.

Table 18-3 Group Entry Schema Mapping between Directory Server and Windows Servers

Directory Server	Active Directory	Windows NT4 Server
cn	**name**	
ntGroupAttributes	groupAttributes	grpi_attributes
ntGroupId	cn name samAccountName	grpi_name
ntGroupType	groupType	
uniqueMember	member	grpi_member

Table 18-4 Group Entry Schema that are directly mapped between Directory Server and Windows Servers

seeAlso	l
description	ou

Manually Initiating Synchronization

While synchronization will always occur automatically every five minutes, you can manually perform synchronization or an update if you have an immediate need for synchronization to occur.

To perform an update manually, go to the Configuration tab in the Console, right-click on the synchronization agreement icon, and select “Send and Receive Updates Now” from the drop down menu.

If you want to re-synchronize every entry in the Directory Server and Windows server, go to the Configuration tab in the Console, right-click on the synchronization agreement icon, and select “Initiate re-synchronization” from the drop down menu. This will not delete or overwrite your data on the sync peer; it will send and receive all updates and add any new or modified Directory Server entries (such as a pre-existing Directory Server user that had the `ntUser` object class added to it).

The Need for Re-Synchronization

As discussed above, entries are synchronized provided they are subject to the sync agreement. There are some cases where an entry can be initially not subject to the agreement (for example, if it lacks the `ntUser` object class) but subsequently becomes subject to the agreement (if the `ntUser` object class is added, for example).

In cases like these, the Directory Server is not able to identify the entry’s change in status with normal update processing, and it will fail to create the corresponding entry in the Windows Server (in the case that that entry does not already exist).

It is necessary to initiate a re-synchronization in order for such entries to be recognized and synchronized. Re-synchronization need only be performed once to identify all such entries.

Checking Synchronization Status

You can check synchronization status in the Status tab of the Console. Highlight the synchronization agreement you want to monitor, and the relevant information should appear in the right-hand pane. From the Status area, you can determine whether the last incremental and total updates were successful and when they occurred. Total update signifies the time when the last re-initializiation operation completed.

Modifying the Synchronization Agreement

It is possible to modify parts of the synchronization agreement after it has been created.

In the Configuration>Replication tab of the Directory Server Console, select the sync agreement icon from beneath the database. There are two tabs: Summary, and Connection.

- The Summary tab allows you to change the description of the agreement. This tab also shows the sync peer host and port information and synchronized subtrees.
- The Connection tab will let you change the bind DN and bind credentials for the sync manager. It will also show whether this is over an SSL connection. Finally, it shows whether new user and group entries will be created in the Directory Server.

Active Directory Schema Compatibility

Although Active Directory supports the same basic X.500 object classes as Directory Server, there are a few subtle incompatibilities of which administrators should be aware:

- Both Active Directory and Directory Server can enforce *password policy* that can enforce certain requirements upon passwords: minimum length, maximum age and so forth. Windows Sync does not synchronize the policies, nor does it ensure that the policies are consistent. This is something that the administrators of both systems must ensure is done. If password policy is not consistent, then password changes made on one system may fail when replayed on the other system.
- Nested groups (where a group contains another group as a member) are supported and will be synchronized. However, Active Directory imposes certain constraints for the composition of nested groups. For example, a domain local group may not be a member of a global group. Directory Server has no concept of local and global groups, and therefore, it is possible to create entries on the Directory Server side that will violate Active Directory's constraints when synchronized. Again, it is the responsibility of the administrators to ensure that this does not happen.
- Active Directory uses the attribute `streetAddress` for a user or group's physical or postal address. Directory Server uses the RFC2798 inetOrgPerson attribute `street` for this purpose. However, as defined in RFC2256, `streetAddress` is an *alias* for `street`. To compound the confusion, Active Directory also has the `street` attribute, but it is not an alias for

`streetAddress` but a separate attribute that can hold an independent value. Windows Sync maps `streetAddress` in Windows to `street` in Directory Server, and therefore, precludes the use of the `street` attribute in Active Directory.

NT4-Specific Limitiations

The NT4 LDAP Service attempts to reflect the NT4 NTLM user database (as accessed via the Net API) in LDAP. In general, this works well, but there are some fundamental incompatibilities between LDAP schema and the underlying data store. These incompatibilities are listed below:

- The schema supported by the NTLM database is severely limited compared to Active Directory. There is little support for information beyond username and full name. The missing attributes therefore cannot be synchronized.
- There is no support for the incremental Dirsync found in Active Directory. What this means is that every time the Directory Server performs a synchronization pass, it will pull the complete set of all entries from NT4. This has implications for the consistency of data because if a modification is made to an entry on the Directory Server side and the same entry is read from NT4 in a synchronization operation before the change has been propagated outbound, then the change will be undone.
- There is no support for tombstone entries in NT4. What this means is that entries deleted from NT4 will not be automatically deleted from the Directory Server side. It will be necessary to delete those entries manually.
- NT4 has no `surname` attribute. However, the `inetOrgPerson` object class requires `surname` have a value. In order to allow the use of the standard person schema with NT4, when new user entries are created in the sync process, they are given a `surname` attribute value that is equal to the NT user name. This can be changed later by the admistrator to the correct value. This issue only applies to new entries created in Directory Server by a sync operation. If the associated Directory Server entry for an NT4 user account already exists, its `surname` attribute is left unchanged.

Troubleshooting

If synchronization does not seem to be functioning properly, see the Windows event log and/or Directory Server error log for information on any potential problems.

You can also enable replication logging for more detailed information on synchronization to be recorded in the error logs:

1. In the Console, click the Configuration tab, select Logs from the navigation menu on the right, and open the Error Log.
2. Scroll down to error log level, and select Replication from the menu. Hit save.

   For complete information on error log levels, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

This will produce verbose logging from the sync code that can help in diagnosing problems.

Error #1 -- NTDS: javax.naming.NamingException: can't invoke ssl filter

There is a problem with the location of your keystore, the keystore filename, the path is not escaped correctly, or you have configured wrong keystore password in the `usersync.conf` file.

Error #2 -- NTDS: org.apache.ldap.common.exception.LdapConfigurationException: Failed to bind the LDAP protocol service to the service registry: (SOCKET, ldap, 0.0.0.0/0.0.0.0:389) [Root exception is java.net.BindException: Address already in use: bind]

The port you are attempting to use is already in use.

Error #3 -- NTDS: javax.naming.NamingException: ERROR: Admin password not set. Server not starting for security reasons.

It is mandatory to set a password for the admin account for the NT4 LDAP service. There is no default password.

Error #4 -- NT4 and Active Directory: The message box when creating the sync agreement indicates that the it cannot connect to Active Directory.

Make sure that the directory suffixes, Windows domain and domain host, and the administrator DN and password are correct. Also verify that the port numbers used for LDAPS is correct. If all of this is correct, make sure that Active Directory or teh Windows machine are running.

Error #5 -- NT4 and Active Directory: After synchronization, the status shows error 81.

One of the sync peer servers has not been properly configured for SSL communication. Examine the Directory Server access log file to see if the connection attempt was received by the Directory Server. You may also find helpful messages in the Directory Server's error log file.

To narrow down the source of the misconfiguration, try to establish an LDAPS connection to the Directory Server. If this connection attempt fails, check all values (port number, hostname, search base, and so forth) to see if any of these are the problem. If all else fails, reconfigure the Directory Server with a new certificate.

If the LDAPS connection is successful, it is likely that the misconfiguration is on the Windows server. Check that you have properly configured the NT4 LDAP Service and Password Sync for SSL and that the NT4 LDAP Service is running. Examine the Windows event log file for error messages.

NOTE A common problem is to fail to trust your certificate authority when configuring Windows Sync service's certificates or to fail to trust the Active Directory or NT4 Server certificate authority in the Directory Server.

Part 3

Appendixes

Appendix A

LDAP Data Interchange Format

Red Hat Directory Server (Directory Server) uses the LDAP Data Interchange Format (LDIF) to describe a directory and directory entries in text format. LDIF is commonly used to build the initial directory database or to add large numbers of entries to the directory all at once. In addition, LDIF is also used to describe changes to directory entries. For this reason, most of Directory Server's command-line utilities rely on LDIF for either input or output.

Because LDIF is a text file format, you can create your LDIF files using virtually any language. All directory data is stored using the UTF-8 encoding of Unicode. Therefore, the LDIF files you create must also be UTF-8 encoded.

This chapter provides information about LDIF in the following sections:

- LDIF File Format (page 575)
- Specifying Directory Entries Using LDIF (page 579)
- Defining Directories Using LDIF (page 583)
- Storing Information in Multiple Languages (page 586)

For information on using LDIF to modify directory entries, see chapter 2, "Creating Directory Entries."

LDIF File Format

LDIF consists of one or more directory entries separated by a blank line. Each LDIF entry consists of an optional entry ID, a required distinguished name, one or more object classes, and multiple attribute definitions.

The LDIF format is defined in RFC 2849, *The LDAP Data Interchange Format (LDIF)*. Directory Server is compliant with this standard.

The basic form of a directory entry represented in LDIF is as follows:

```
dn: distinguished_name
objectClass: object_class
objectClass: object_class
...
attribute_type[;subtype]:attribute_value
attribute_type[;subtype]:attribute_value
...
```

You must supply the DN and at least one object class definition. In addition, you must include any attributes required by the object classes that you define for the entry. All other attributes and object classes are optional. You can specify object classes and attributes in any order. The space after the colon is also optional. For information on standard object classes and attributes, refer to *Red Hat Directory Server Schema Reference*.

Table A-1 describes the LDIF fields shown in the previous definition.

Table A-1 LDIF Fields

Field	**Definition**
[*id*]	*Optional*. A positive decimal number representing the entry ID. The database creation tools generate this ID for you. Never add or edit this value yourself.
`dn:` *distinguished_name*	Specifies the distinguished name for the entry. For a complete description of distinguished names, refer to the *Red Hat Directory Server Deployment Guide*.
`objectClass:` *object_class*	Specifies an object class to use with this entry. The object class identifies the types of attributes, or schema, allowed and required for the entry. See the *Red Hat Directory Server Schema Reference* for a list of standard object classes and chapter 9, "Extending the Directory Schema," for information on customizing the schema.
attribute_type	Specifies a descriptive attribute to use with the entry. The attribute should be defined either in the schema. See the *Red Hat Directory Server Schema Reference* for a list of standard attributes and chapter 9, "Extending the Directory Schema," for information on customizing the schema.

Table A-1 LDIF Fields *(Continued)*

Field	Definition
[*subtype*]	*Optional.* Specifies subtype, language, binary, or pronunciation. Use this tag to identify the language in which the corresponding attribute value is expressed or whether the attribute value is binary or a pronunciation of an attribute value. For information on attribute subtypes, see "Adding an Attribute Subtype," on page 55. For a complete list of the supported subtypes tags, see Table D-2, on page 619.
attribute_value	Specifies the attribute value to be used with the attribute type.

The LDIF syntax for representing a change to an entry in the directory is different from the syntax described above. For information on using LDIF to modify directory entries, see chapter 2, "Creating Directory Entries."

Continuing Lines in LDIF

When you specify LDIF, you can break and continue, or fold, a line by indenting the continued portion of the line by exactly one space. For example, the following two statements are identical:

```
dn: cn=Jake Lupinski,dc=example,dc=com
```

```
dn: cn=Jake Lup
 inski, dc=exa
 mple,dc=com
```

You are not required to break and continue LDIF lines. However, doing so may improve the readability of your LDIF file.

Representing Binary Data

You can represent binary data, such as a JPEG image, in LDIF using one of the following methods:

- The standard LDIF notation, the lesser than (<) symbol. For example:

  ```
  jpegphoto: < file:/path/to/photo
  ```

If you use this standard notation, you do not need to specify the `ldapmodify -b` parameter. However, you must add the following line to the beginning of your LDIF file or your LDIF update statements:

```
version:1
```

For example, you could use the following `ldapmodify` command:

```
prompt> ldapmodify -D userDN -w user_password

>version: 1
>dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate
userCertificate;binary:< file: BarneysCert
```

- Using base 64-encoding. You identify base 64-encoded data by using the `::` symbol. For example:

  ```
  jpegPhoto:: encoded_data
  ```

In addition to binary data, other values that must be base 64-encoded include:

- Any value that begins with a colon (:) or a space.
- Any value that contains non-ASCII data, including new lines.

Use the `ldif` command-line utility with the `-b` parameter to convert binary data to LDIF format:

```
ldif -b attribute_name
```

where *attribute_name* is the name of the attribute to which you are supplying the binary data. The binary data is read from standard input and the results are written to standard output. Thus, you should use redirection operators to select input and output files.

The `ldif` command-line utility will take any input and format it with the correct line continuation and appropriate attribute information. The `ldif` utility also assesses whether the input requires base 64-encoding. For example:

```
ldif -b jpegPhoto < mark.jpg > out.ldif
```

This example takes a binary file containing a JPEG-formatted image and converts it into LDIF format for the attribute named `jpegPhoto`. The output is saved to `out.ldif`.

The `-b` option specifies that the `ldif` utility should interpret the entire input as a single binary value. If `-b` is not present, each line is considered to be a separate input value.

Specifying Directory Entries Using LDIF

You can store many types of entries in your directory. This section concentrates on three of the most common types of entries used in a directory: organization, organizational unit, and organizational person entries.

The object classes defined for an entry are what indicate whether the entry represents an organization, an organizational unit, an organizational person, or some other type of entry. For a general discussion of the types of entries you can create in your directory, see the *Red Hat Directory Server Deployment Guide*. For a complete list of the object classes you can use by default in your directory and a list of the most commonly used attributes, see the *Red Hat Directory Server Schema Reference*.

Specifying Organization Entries

Directories often have at least one organization entry. Typically this is the first, or topmost, entry in your directory. The organization entry often corresponds to the suffix set for your directory. For example, if your directory is defined to use a suffix of `dc=example,dc=com`, then you will probably have an organization entry in your directory named `dc=example,dc=com`.

The LDIF that you specify to define an organization entry should appear as follows:

```
dn: distinguished_name
objectClass: top
objectClass: organization
o: organization_name
list_of_optional_attributes
...
```

The following is a sample organization entry in LDIF format:

```
dn: dc=example,dc=com
objectclass: top
objectclass: organization
o: example.com Corporation
description: Fictional company for example purposes
telephonenumber: 555-5555
```

The organization name in the following example uses a comma:

```
dn: o="example.com Chile\\, S.A."
objectclass: top
objectclass: organization
o: "example.com Chile\\, S.A."
description: Fictional company for example purposes
telephonenumber: 555-5556
```

Each element of the LDIF-formatted organization entry is defined in Table A-2.

Table A-2 LDIF Elements in Organization Entries

LDIF Element	Description
`dn:` *distinguished_name*	Specifies the distinguished name for the entry. DNs are described in the *Red Hat Directory Server Deployment Guide*. A DN is required.
`objectClass: top`	*Required.* Specifies the `top` object class.
`objectClass: organization`	Specifies the `organization` object class. This line defines the entry as an organization. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.
`o:` *organization_name*	Attribute that specifies the organization's name. If the organization name includes a comma, you must escape the comma by a single backslash, and the entire organization argument must be enclosed in quotation marks. If you are working with a UNIX shell, this backslash will also need escaping, which means that you will have to use two backslashes. For example, to set the suffix to `example.com Bolivia, S.A.` you would enter `"o: example.com Bolivia\\, S.A."`.
list_of_attributes	Specifies the list of optional attributes that you want to maintain for the entry. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Specifying Organizational Unit Entries

Organizational unit entries are often used to represent major branch points, or subdirectories, in your directory tree. They correspond to major, reasonably static entities within your enterprise, such as a subtree that contains people or a subtree that contains groups. However, the organizational unit attribute that is contained in the entry may also represent a major organization within your enterprise, such as marketing or engineering.

There is usually more than one organizational unit, or branch point, within a directory tree. For information on how to design your directory tree, see the *Red Hat Directory Server Deployment Guide*.

The LDIF that you specify to define an organizational unit entry must appear as follows:

```
dn: distinguished_name
objectClass: top
objectClass: organizationalUnit
ou: organizational_unit_name
list_of_optional_attributes
...
```

The following is a sample organizational unit entry in LDIF format:

```
dn: ou=people, dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: people
description: Fictional organizational unit for example purposes
```

Table A-3 defines each element of the LDIF-formatted organizational unit entry.

Table A-3 LDIF Elements in Organizational Unit Entries

LDIF Element	Description
`dn:` *distinguished_name*	Specifies the distinguished name for the entry. A DN is required. If there is a comma in the DN, the comma must be escaped with a backslash (\). For example: `dn: ou=people,o=example.com Bolivia\,S.A.`
`objectClass: top`	*Required.* Specifies the `top` object class.
`objectClass: organizationalUnit`	Specifies the `organizationalUnit` object class. This line defines the entry as an organizational unit. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Table A-3 LDIF Elements in Organizational Unit Entries *(Continued)*

LDIF Element	Description
`ou:` *organizational_unit_name*	Attribute that specifies the organizational unit's name.
list_of_attributes	Specifies the list of optional attributes that you want to maintain for the entry. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Specifying Organizational Person Entries

The majority of the entries in your directory represent organizational people.

In LDIF, the definition of an organizational person is as follows:

```
dn: distinguished_name
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: common_name
sn: surname
list_of_optional_attributes
```

The following is an example organizational person entry in LDIF format:

```
dn: uid=bjensen,ou=people,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenname: Babs
uid: bjensen
ou: Marketing
ou: people
description: Fictional person for example purposes
telephonenumber: 555-5557
userpassword: {sha}dkfljlk34r2kljdsfk9
```

Table A-4 defines each aspect of the LDIF person entry.

Table A-4 LDIF Elements in Person Entries

LDIF Element	Description
`dn:` *distinguished_name*	Specifies the distinguished name for the entry. A DN is required. If there is a comma in the DN, the comma must be escaped with a backslash (\). For example, `dn:uid=bjensen,ou=people,o=example.com Bolivia\,S.A.`
`objectClass: top`	*Required.* Specifies the `top` object class.
`objectClass: person`	Specifies the `person` object class. This object class specification should be included because many LDAP clients require it during search operations for a person or an organizational person.
`objectClass: organizationalPerson`	Specifies the `organizationalPerson` object class. This object class specification should be included because some LDAP clients require it during search operations for an organizational person.
`objectClass: inetOrgPerson`	Specifies the `inetOrgPerson` object class. The `inetOrgPerson` object class is recommended for the creation of an organizational person entry because this object class includes the widest range of attributes. The `uid` attribute is required by this object class, and entries that contain this object class are named based on the value of the `uid` attribute. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.
`cn:` *common_name*	Specifies the person's common name, which is the full name commonly used by the person. For example, `cn: Bill Anderson`. At least one common name is required.
`sn:` *surname*	Specifies the person's surname, or last name. For example, `sn: Anderson`. A surname is required.
list_of_attributes	Specifies the list of optional attributes that you maintain for the entry. See the *Red Hat Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Defining Directories Using LDIF

You can define the contents of an entire directory using LDIF. Using LDIF is an efficient method of directory creation when you have many entries to add to the directory.

To create a directory using LDIF, follow these steps:

1. Create an ASCII file containing the entries you want to add in LDIF format.

   Make sure each entry is separated from the next by an empty line. You should use just one line, and the first line of the file must not be blank, or else the `ldapmodify` utility will exit. For more information, see "Specifying Directory Entries Using LDIF," on page 579.

2. Begin each file with the topmost, or root, entry in the database.

   The root entry must represent the suffix or sub-suffx contained by the database. For example, if your database has the suffix `dc=example,dc=com`, the first entry in your directory must be

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

   For information on suffixes, see the "Suffix" parameter described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

3. Make sure that an entry representing a branch point in the LDIF file is placed before the entries that you want to create under that branch.

   For example, if you want to place an entry in a people and a group subtree, create the branch point for those subtrees before creating entries within those subtrees.

4. Create the directory from the LDIF file using one of the following methods:

   - Directory Server Console

     Use this method if you have a small database to import (less than 1000 entries). See "Importing a Database from the Console," on page 152.

   - `ldif2db` command-line utility

     Use this method if you have a large database to import (more than 1,000 entries). See "Importing Using the ldif2db Command-Line Script," on page 155.

   - `ldapmodify` command-line utility with the `-a` parameter

     Use this method if you currently have a directory database, but you are adding a new subtree to the database. Unlike the other methods for creating the directory from an LDIF file, Directory Server must be running before you can add a subtree using `ldapmodify`. See "Adding and Modifying Entries Using ldapmodify," on page 60.

LDIF File Example

The following example shows an LDIF file that contains one organization, two organizational units, and three organizational person entries:

```
dn: o=example.com Corp,dc=example,dc=com
objectclass: top
objectclass: organization
o: example.com Corp
description: Fictional organization for example purposes

dn: ou=People,o=example.com Corp,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People
description: Fictional organizational unit for example purposes
tel: 555-5559

dn: cn=June Rossi,ou=People,o=example.com Corp,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: June Rossi
sn: Rossi
givenName: June
mail: rossi@example.com
userPassword: {sha}KDIE3AL9DK
ou: Accounting
ou: people
telephoneNumber: 2616
roomNumber: 220

dn: cn=Marc Chambers,ou=People,o=example.com
Corp,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Marc Chambers
sn: Chambers
givenName: Marc
mail: chambers@example.com
userPassword: {sha}jdl2alem87dlacz1
telephoneNumber: 2652
ou: Manufacturing
ou: People
roomNumber: 167
```

```
dn: cn=Robert Wong,ou=People,example.com Corp,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Robert Wong
cn: Bob Wong
sn: Wong
givenName: Robert
givenName: Bob
mail: bwong@example.com
userPassword: {sha}nn2msx761
telephoneNumber: 2881
roomNumber: 211
ou: Manufacturing
ou: people

dn: ou=Groups,o=example.com Corp,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups
description: Fictional organizational unit for example purposes
```

Storing Information in Multiple Languages

If your directory contains a single language, you do not need to do anything special to add a new entry to the directory. However, if your organization is multinational, you may find it necessary to store information in multiple languages so that users in different locales can view directory information in their own language.

When information in your directory is represented in multiple languages, the server associates language tags with attribute values. When you add a new entry, you must provide attribute values used in the RDN (Relative Distinguished Name) without any language codes.

You can even store multiple languages within a single attribute. When you do, the attribute types are the same, but each value has a different language code.

For a list of the languages supported by Directory Server and their associated language tags, see “Identifying Supported Locales,” on page 617.

NOTE The language tag has no effect on how the string is stored within the directory. All object class and attribute strings are stored using UTF-8.

For example, suppose `example.com` Corporation has offices in the United States and France and wants employees to be able to view directory information in their native language. When adding directory entries, the directory administrator chooses to provide attribute values in both English and French. When adding a directory entry for a new employee, Babs Jensen, the administrator creates the following LDIF entry:

```
dn: uid=bjensen,ou=people,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
name: Babs Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
streetAddress: 1 University Street
streetAddress;lang-en: 1 University Street
streetAddress;lang-fr: 1 rue de l'Université
preferredLanguage: fr
```

Users accessing this directory entry with an LDAP client with the preferred language set to English will see the address `1 University Street`. Users accessing the directory with an LDAP client with the preferred language set to French will see the address `1 rue de l'Université`.

Appendix B

Finding Directory Entries

You can find entries in your directory using any LDAP client. Most clients provide some form of a search interface that allows you to search the directory easily and retrieve entry information.

NOTE You cannot search the directory unless the appropriate access control has been set in your directory. For information on setting access control in your directory, see chapter 6, "Managing Access Control."

This chapter covers the following topics:

- Finding Entries Using the Server Console (page 589)
- Using ldapsearch (page 590)
- LDAP Search Filters (page 597)
- Searching an Internationalized Directory (page 601)

Finding Entries Using the Server Console

Use the Directory tab of the Directory Server Console to browse the contents of the directory tree and search for specific entries in the directory.

1. Make sure the Directory Server is running.
2. Start Directory Server Console.

   See "Starting Directory Server Console," on page 34, for specific instructions.

3. On Directory Server Console, select the Directory tab.

   Depending on the DN you used to authenticate to the directory, this tab displays the contents of the directory that you have access permissions to view. You can browse through the contents of the tree, or right-click an entry, and select Search from the pop-up menu. See the online help for information on using this feature.

CAUTION Do not modify the contents of the `o=NetscapeRoot` suffix using the Directory tab unless instructed to do so by Red Hat Technical Support.

Using ldapsearch

You can use the `ldapsearch` command-line utility to locate and retrieve directory entries. This utility opens a connection to the specified server using the specified distinguished name and password and locates entries based on a specified search filter. Search scopes can include a single entry, an entry's immediate subentries, or an entire tree or subtree.

Search results are returned in LDIF format.

This section contains information about the following topics:

- Using Special Characters
- ldapsearch Command-Line Format
- Commonly Used ldapsearch Options
- ldapsearch Examples

Using Special Characters

When using the `ldapsearch` command-line utility, you may need to specify values that contain characters that have special meaning to the command-line interpreter (such as space [ ], asterisk [*], backslash [\], and so forth). When you specify special characters, enclose the value in quotation marks (""). For example:

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

Depending on your command-line interpreter, use either single or double quotation marks for this purpose. Refer to your operating system documentation for more information.

ldapsearch Command-Line Format

When you use `ldapsearch`, you must enter the command using the following format:

```
ldapsearch [optional_options] [optional_search_filter] [optional_list_of_attributes]
```

where

- *optional_options* represents a series of command-line options. These must be specified before the search filter, if any.
- *optional_search_filter* represents an LDAP search filter as described in "LDAP Search Filters," on page 597. Do not specify a search filter if you are supplying search filters in a file using the `-f` option.
- *optional_list_of_attributes* represents 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 "Displaying Subsets of Attributes," on page 595. If you do not specify a list of attributes, the search returns values for all attributes permitted by the access control set in the directory (with the exception of operational attributes).

NOTE If you want operational attributes returned as a result of a search operation, you must explicitly specify them 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.

Commonly Used ldapsearch Options

The following table lists the most commonly used `ldapsearch` command-line options. If you specify a value that contains a space [ ], the value should be surrounded by double quotation marks; for example,
`-b "ou=groups, dc=example,dc=com"`.

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 option 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 double quotation marks. For example:
	
	`-b "cn=Barbara Jensen, ou=Product Development, dc=example,dc=com"`
	
	If you want to search the root DSE entry, specify an empty string here. For example:
	`-b ""`

`-D`	Specifies the distinguished name with which to authenticate to the server. This option is optional if anonymous access is supported by your 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"`
`-h`	Specifies the hostname or IP address of the machine on which the Directory Server is installed. If you do not specify a host, `ldapsearch` uses the localhost. For example, `-h mozilla`.
`-l`	Specifies the maximum number of seconds to wait for a search request to complete. Regardless of the value specified here, `ldapsearch` will never wait longer than is allowed by the server's `nsslapd-timelimit` attribute. For example, `-l 300`. The default value for the `nsslapd-timelimit` attribute is `3600` seconds.
`-p`	Specifies the TCP port number that the Directory Server uses. For example, `-p 1049`. The default is `389`. If `-Z` is used, the default is `636`.

Option	Description
`-s`	Specifies the scope of the search. The scope can be one of the following: • `base` — Search only the entry specified in the `-b` option or defined by the `LDAP_BASEDN` environment variable. • `one` — Search 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` — Search 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.
`-w`	Specifies the password associated with the distinguished name that is specified in the `-D` option. If you do not specify this option, anonymous access is used. For example, `-w diner892`.
`-x`	Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server rather than on the client.
`-z`	Specifies 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, you can override this limitation by binding as the root DN when using this command-line argument. When you bind as the root DN, this option defaults to zero (`0`). The default value for the `nsslapd-sizelimit` attribute is `2000` entries.

For detailed information on all `ldapsearch` utility options, refer to the *Red Hat Directory Server Configuration, Command, and File Reference*.

ldapsearch Examples

In the next set of examples, suppose the following are true:

- You want to perform a search of all entries in the directory.
- You have configured your directory to support anonymous access for search and read. You do not have to specify any bind information in order to perform the search. For more information on anonymous access, see "Defining User Access - userdn Keyword," on page 225.
- The server is located on hostname `mozilla`.
- The server uses port number `389`. Since this is the default port, you do not have to identify the port number on the search request.
- SSL is enabled for the server on port `636` (the default SSL port number).
- The suffix under which all data is stored is `dc=example,dc=com`.

Returning All Entries

Given the previous information, the following call will return all entries in the directory:

```
ldapsearch -h mozilla -b "dc=example,dc=com" -s sub
"objectclass=*"
```

`"objectclass=*"` is a search filter that matches any entry in the directory.

Specifying Search Filters on the Command-Line

You can specify a search filter directly on the command-line. If you do this, be sure to enclose your filter in quotation marks ("filter"). Also, do not specify the `-f` option. For example:

```
ldapsearch -h mozilla -b "dc=example,dc=com" "cn=babs jensen"
```

Searching the Root DSE Entry

The root DSE is a special entry that contains a list of all the suffixes supported by the local Directory Server. You can search this entry by supplying a search base of "". You must also specify a search scope of `base` and a filter of `"objectclass=*"`. For example:

```
ldapsearch -h mozilla -b "" -s base "objectclass=*"
```

Searching the Schema Entry

Directory Server stores all directory server schema in the special `cn=schema` entry. This entry contains information on every object class and attribute defined for your Directory Server.

You can examine the contents of this entry as follows:

```
ldapsearch -h mozilla -b "cn=schema" -s base "objectclass=*"
```

Using LDAP_BASEDN

To make searching easier, you can set your search base using the `LDAP_BASEDN` environment variable. Doing this allows you to skip specifying the search base with the `-b` option. For information on how to set environment variables, see the documentation for your operating system.

Typically, you set `LDAP_BASEDN` to your directory's suffix value. Since your directory suffix is equal to the root, or topmost, entry in your directory, this causes all searches to begin from your directory's root entry.

For example, suppose you have set `LDAP_BASEDN` to `dc=example,dc=com`. Then to search for `cn=babs jensen` in your directory, use the following command-line call:

```
ldapsearch -h mozilla "cn=babs jensen"
```

In this example, the default scope of `sub` is used because the `-s` option was not used to specify the scope.

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 you are allowed to read. You can set up the directory access control such that you are allowed to read only a subset of the attributes on any given directory entry. Only operational attributes are not returned. If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command.

Suppose you do not want to see all of the attributes returned in the search results. You can limit the returned attributes to just a few specific attributes by specifying the ones you want 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 -h mozilla "objectclass=*" sn cn
```

This example assumes you set your search base with `LDAP_BASEDN`.

Specifying Search Filters Using a File

You can enter search filters into a file instead of entering them on the command-line. When you do this, 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, if the file contains:

```
sn=Francis
givenname=Richard
```

then `ldapsearch` first finds all the entries with the surname `Francis` and, 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, suppose you specified the previous search filters in a file named `searchdb`, and you set your search base using `LDAP_BASEDN`. Then the following returns all the entries that match either search filter:

```
ldapsearch -h mozilla -f searchdb
```

You can limit the set of attributes returned here by specifying the attribute names that you want 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 -h mozilla -f searchdb sn givenname
```

Specifying DNs That Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, you must escape the comma with a backslash (\). For example, to find everyone in the `example.com Bolivia, S.A.` subtree, use the following command:

```
ldapsearch -h mozilla -s base -b "o=example.com Bolivia\,
S.A.,dc=example,dc=com" "objectclass=*"
```

Using Client Authentication When Searching

This example shows user `bjensen` searching the directory using client authentication:

```
ldapsearch -h mozilla -p 636 -b "dc=example,dc=com" -N
"bjensenscertname" -Z -W certdbpassword -P
/home/bjensen/certdb/cert.db "givenname=Richard"
```

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 you use `ldapsearch`, you can place multiple search filters in a file, with each filter on a separate line in the file, or you can specify a search filter directly on the command-line.

For example, the following filter specifies a search for the common name Babs Jensen:

```
cn=babs jensen
```

This search filter returns all entries that contain the common name Babs Jensen. Searches for common name values are not case sensitive.

When the common name attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match this filter:

```
cn: babs jensen
cn;lang-fr: babs jensen
```

For a list of all the supported language tags, see Table D-1, on page 617.

Search Filter Syntax

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. You can also define filters that use different attributes combined together with Boolean operators.

Search filters are described in detail in the following sections:

- Using Attributes in Search Filters
- Using Operators in Search Filters
- Using Compound Search Filters
- Search Filter Examples

Using Attributes in Search Filters

When searching for an entry, you can specify attributes associated with that type of entry. For example, when you search for people entries, you can use the `cn` attribute to search for people with a specific common name.

Examples of attributes that people entries might include:

- `cn` — the person's common name.
- `sn` — the person's surname, or last name, or family name.
- `telephoneNumber` — the person's telephone number.
- `buildingName` — the name of the building in which the person resides.
- `l` — the locality where you can find the person.

For a listing of the attributes associated with types of entries, see the *Red Hat Directory Server Schema Reference*.

Using Operators in Search Filters

The operators that you can use in search filters are listed in Table B-1. In addition to these search filters, you can specify special filters to work with a preferred language collation order. For information on how to search a directory with international charactersets, see "Searching an Internationalized Directory," on page 601.

Table B-1 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`

Table B-1 Search Filter Operators *(Continued)*

Search Type	Operator	Description
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`

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*) . . .)

where *Boolean-operator* is any one of the Boolean operators listed in Table B-2.

Boolean operators can be combined and nested together to form complex expressions, such as:

(*Boolean-operator* (*filter*) ((*Boolean-operator* (*filter*) (*filter*))))

The Boolean operators available for use with search filters include the following:

Table B-2 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)...)`

Table B-2 Search Filter Boolean Operators *(Continued)*

Operator	Symbol	Description
`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.

Search Filter Examples

The following filter searches for entries containing one or more values for the manager attribute. This is also known as a presence search:

```
manager=*
```

The following filter searches for entries containing the common name `Ray Kultgen`. This is also known as an equality search:

```
cn=Ray Kultgen
```

The following filter returns all entries that do not contain the common name `Ray Kultgen`:

```
(!(cn=Ray Kultgen))
```

The following filter returns all entries that contain a description attribute that contains the substring `X.500`:

```
description=*X.500*
```

The following 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*)))
```

The following filter returns all entries whose organizational unit is `Marketing` and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(|(manager=cn=Julie
Fulmer,ou=Marketing,dc=example,dc=com)(manager=cn=Cindy
Zwaska,ou=Marketing,dc=example,dc=com)))
```

The following filter returns all entries that do not represent a person:

```
(!(objectClass=person))
```

The following filter returns all entries that do not represent a person and whose common name is similar to `printer3b`:

```
(&(!(objectClass=person))(cn~=printer3b))
```

Searching an Internationalized Directory

When you perform search operations, you can request that the directory sort the results based on any language for which the server has a supporting collation order. For a listing of the collation orders supported by the directory, see "Identifying Supported Locales," on page 617.

NOTE When performing internationalized searches, you must perform an LDAP v3 search; therefore, do not specify the `-V2` option on the call to `ldapsearch`.

This section focuses on the matching rule filter portion of the `ldapsearch` syntax. For more information on general `ldapsearch` syntax, see "LDAP Search Filters," on page 597. For information on searching internationalized directories using the Users and Groups portion of the Red Hat Console, refer to the online help or *Managing Servers with Red Hat Console*.

This section covers the following topics:

- Matching Rule Filter Syntax
- Supported Search Types
- International Search Examples

Matching Rule Filter Syntax

A matching rule provides special guidelines for how the directory compares strings during a search operation. In an international search, the matching rule tells the system what collation order and operator to use when performing the search operation. For example, a matching rule in an international search might tell the server to search for attribute values that come at or after llama in the Spanish collation order. The syntax of the matching rule filter is as follows:

attr:*matchingRule*:=*value*

where:

- *attr* is an attribute belonging to entries you're searching for, such as `cn` or `mail`.
- *matchingRule* is a string that identifies either the collation order or the collation order and a relational operator, depending on the format you prefer. For a discussion of matching rule formats, see "Matching Rule Formats," on page 602.
- *value* is either the attribute value you want to search for or a relational operator plus the attribute value you want to search for.The syntax of the value portion of the filter depends on the matching rule format you use.

Matching Rule Formats

The matching rule portion of a search filter can be represented in several ways. The one you use is a matter of personal preference. The matching rule can be represented in the following ways:

- As the OID of the collation order for the locale on which you want to base your search.
- As the language tag associated with the collation order on which you want to base your search.
- As the OID of the collation order and a suffix that represents a relational operator.
- As the language tag associated with the collation order and a suffix that represents a relational operator.

The syntax for each of these options is discussed in the following sections:

- Using an OID for the Matching Rule
- Using a Language Tag for the Matching Rule
- Using an OID and Suffix for the Matching Rule
- Using a Language Tag and Suffix for the Matching Rule

Using an OID for the Matching Rule

Each locale supported by the directory server has an associated collation order OID. For a list of locales supported by the directory server and their associated OIDs, see Table D-1, on page 617.

You can use the collation order OID in the matching rule portion of the matching rule filter as follows:

attr:*OID*:=(*relational_operator value*)

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search for all `departmentNumber` attributes that are at or after `N4709` in the Swedish collation order, use the following filter:

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

Using a Language Tag for the Matching Rule

Each locale supported by the directory server has an associated language tag. For a list of locales supported by the directory server and their associated language tags, see Table D-1, on page 617.

You can use the language tag in the matching rule portion of the matching rule filter as follows:

attr:*language-tag*:=(*relational_operator value*)

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search the directory for all description attributes with a value of `estudiante` using the Spanish collation order, use the following filter:

```
cn:es:== estudiante
```

Using an OID and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the OID in the matching rule portion of the filter. Combine the OID and suffix as follows:

attr:*OID+suffix*:=*value*

For example, to search for `businessCategory` attributes with the value `softwareprodukte` in the German collation order, use the following filter:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
```

The `.3` in the previous example is the equality suffix.

For a list of locales supported by the directory server and their associated OIDs, see Table D-1, on page 617. For a list of relational operators and their equivalent suffixes, see Table B-3, on page 605.

Using a Language Tag and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the language tag in the matching rule portion of the filter. Combine the language tag and suffix as follows:

attr:*language-tag+suffix*:=*value*

For example, to search for all surnames that come at or after `La Salle` in the French collation order, use the following filter:

```
sn:fr.4:=La Salle
```

For a list of locales supported by the directory server and their associated language tags, see Table D-1, on page 617. For a list of relational operators and their equivalent suffixes, see Table B-3, on page 605.

Using Wildcards in Matching Rule Filters

When performing a substring search using a matching rule filter, you can 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`, you would enter a `l*n` in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter `u`, you would enter a value of `u*` in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, you must escape the * 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\2a*Net product line
```

Supported Search Types

The directory server supports the following types of international searches:

- equality (=)
- substring (*)
- greater-than (>)
- greater-than or equal-to (>=)
- less-than (<)
- less-than or equal-to (<=)

Approximate, or phonetic, and presence searches are supported only in English.

As with a regular `ldapsearch` search operation, an international search uses operators to define the type of search. However, when invoking an international search, you can either use the standard operators (=, >=, >, <, <=) in the value portion of the search string, or you can use a special type of operator, called a suffix (not to be confused with the directory suffix), in the matching rule portion of the filter. Table B-3 summarizes each type of search, the operator, and the equivalent suffix.

Table B-3 Search Types, Operators, and Suffixes

Search Type	Operator	Suffix
Less-than	`<`	`.1`
Less-than or equal-to	`<=`	`.2`
Equality	`=`	`.3`
Greater-than or equal-to	`>=`	`.4`
Greater-than	`>`	`.5`
Substring	`*`	`.6`

International Search Examples

The following sections show examples of how to perform international searches on directory data. Each example gives all the possible matching rule filter formats so that you can become familiar with the formats and select the one that works best for you.

Less-Than Example

When you perform a locale-specific search using the less-than operator (<), or suffix (`.1`), you search for all attribute values that come before the given attribute in a specific collation order.

For example, to search for all surnames that come before the surname Marquez in the Spanish collation order, you could use any of the following matching rule filters:

```
sn:2.16.840.1.113730.3.3.2.15.1:=< Marquez
sn:es:=< Marquez
sn:2.16.840.1.113730.3.3.2.15.1.1:=Marquez
sn:es.1:=Marquez
```

Less-Than or Equal-to Example

When you perform a locale-specific search using the less-than or equal-to operator (<=), or suffix (.2), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all room numbers that come at or before room number CZ422 in the Hungarian collation order, you could use any of the following matching rule filters:

```
roomNumber:2.16.840.1.113730.3.3.2.23.1:=<= CZ422
roomNumber:hu:=<= CZ422
roomNumber:2.16.840.1.113730.3.3.2.23.1.2:=CZ422
roomNumber:hu.2:=CZ422
```

Equality Example

When you perform a locale-specific search using the equal to operator (=), or suffix (.3), you search for all attribute values that match the given attribute in a specific collation order.

For example, to search for all `businessCategory` attributes with the value `softwareprodukte` in the German collation order, you could use any of the following matching rule filters:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1:==
softwareprodukte
businessCategory:de:== softwareprodukte
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
businessCategory:de.3:=softwareprodukte
```

Greater-Than or Equal-to Example

When you perform a locale-specific search using the greater-than or equal-to operator (>=), or suffix (.4), you search for all attribute values that come at or after the given attribute in a specific collation order.

For example, to search for all localities that come at or after Québec in the French collation order, you could use any of the following matching rule filters:

```
locality:2.16.840.1.113730.3.3.2.18.1:=>= Québec
locality:fr:=>= Québec
locality:2.16.840.1.113730.3.3.2.18.1.4:=Québec
locality:fr.4:=Québec
```

Greater-Than Example

When you perform a locale-specific search using the greater-than operator (>), or suffix (`.5`), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all mail hosts that come after host `schranka4` in the Czechoslovakian collation order, you could use any of the following matching rule filters:

```
mailHost:2.16.840.1.113730.3.3.2.5.1:=> schranka4
mailHost:cs:=> schranka4
mailHost:2.16.840.1.113730.3.3.2.5.1.5:=schranka4
mailHost:cs.5:=schranka4
```

Substring Example

When you perform an international substring search, you search for all values that match the given pattern in the specified collation order.

For example, to search for all user IDs that end in `ming` in the Chinese collation order, you could use any of the following matching rule filters:

```
uid:2.16.840.1.113730.3.3.2.49.1:=* *ming
uid:zh:=* *ming
uid:2.16.840.1.113730.3.3.2.49.1.6:=* *ming
uid:zh.6:=* *ming
```

Substring search filters that use DN-valued attributes, such as `modifiersName` or `memberOf`, do not always match entries correctly if the filter contains one or more space characters.

To work around this problem, use the entire DN in the filter instead of a substring, or ensure that the DN substring in the filter begins at an RDN boundary; that is, make sure it starts with a "type=" part of the DN. For example, this filter should not be used:

```
(memberof=*Domain Administrators*)
```

But either one of these will work correctly:

```
(memberof=cn=Domain Administrators*)
(memberof=cn=Domain Administrators,ou=Groups,dc=example,dc=com)
```

Appendix C

LDAP URLs

When you access the Red Hat Directory Server (Directory Server) using a web-based client such as Directory Server Gateway, you must provide an LDAP URL identifying the Directory Server you wish to access.

You also use LDAP URLs when managing Directory Server referrals or access control instructions. This appendix contains the following sections:

Components of an LDAP URL

LDAP URLs have the following syntax:

`ldap[s]://`*hostname*`:`*port*`/`*base_dn*`?`*attributes*`?`*scope*`?`*filter*

The `ldap://` protocol is used to connect to LDAP servers over unsecured connections, and the `ldaps://` protocol is used to connect to LDAP servers over SSL connections. Table C-1 lists the components of an LDAP URL.

Table C-1 LDAP URL Components

Component	Description
hostname	Name (or IP address in dotted format) of the LDAP server. For example: `ldap.example.com` or `192.202.185.90`
port	Port number of the LDAP server (for example, `696`). If no port is specified, the standard LDAP port (`389`) or LDAPS port (`636`) is used.

Table C-1 LDAP URL Components *(Continued)*

Component	Description
base_dn	Distinguished name (DN) of an entry in the directory. This DN identifies the entry that is the starting point of the search. If no base DN is specified, the search starts at the root of the directory tree.
attributes	The attributes to be returned. To specify more than one attribute, use commas to separate the attributes (for example, `"cn,mail,telephoneNumber"`). If no attributes are specified in the URL, all attributes are returned.
scope	The scope of the search, which can be one of these values: • `base` retrieves information only about the distinguished name (*base_dn*) specified in the URL. • `one` retrieves information about entries one level below the distinguished name (*base_dn*) specified in the URL. The base entry is not included in this scope. • `sub` retrieves information about entries at all levels below the distinguished name (*base_dn*) specified in the URL. The base entry is included in this scope. If no scope is specified, the server performs a `base` search.
filter	Search filter to apply to entries within the specified scope of the search. If no filter is specified, the server uses the filter `(objectClass=*)`.

The attributes, scope, and filter components are identified by their positions in the URL. If you do not want to specify any attributes, you still need to include the question marks delimiting that field.

For example, to specify a subtree search starting from `"dc=example,dc=com"` that returns all attributes for entries matching `"(sn=Jensen)"`, use the followingLDAP URL:

```
ldap://ldap.example.com/dc=example,dc=com??sub?(sn=Jensen)
```

The two consecutive question marks `??` indicate that no attributes have been specified. Since no specific attributes are identified in the URL, all attributes are returned in the search.

Escaping Unsafe Characters

Any “unsafe” characters in the URL need to be represented by a special sequence of characters. This is called escaping unsafe characters.

For example, a space is an unsafe character that must be represented as `%20` within the URL. Thus, the distinguished name `"o=example.com corporation"` must be encoded as `"o=example.com%20corporation"`.

The following table lists the characters that are considered unsafe within URLs and provides the associated escape characters to use in place of the unsafe character:

Unsafe Character	**Escape Characters**
space	`%20`
<	`%3c`
>	`%3e`
"	`%22`
#	`%23`
%	`%25`
{	`%7b`
}	`%7d`
|	`%7c`
\	`%5c`
^	`%5e`
~	`%7e`
[	`%5b`
]	`%5d`
‘	`%60`

Examples of LDAP URLs

Example 1:
The following LDAP URL specifies a base search for the entry with the distinguished name `dc=example,dc=com`.

```
ldap://ldap.example.com/dc=example,dc=com
```

- Because no port number is specified, the standard LDAP port number (`389`) is used.
- Because no attributes are specified, the search returns all attributes.
- Because no search scope is specified, the search is restricted to the base entry `dc=example,dc=com`.
- Because no filter is specified, the directory uses the default filter (`objectclass=*`).

Example 2:
The following LDAP URL retrieves the `postalAddress` attribute of the entry with the DN `dc=example,dc=com`:

```
ldap://ldap.example.com/dc=example,dc=com?postalAddress
```

- Because no search scope is specified, the search is restricted to the base entry `dc=example,dc=com`.
- Because no filter is specified, the directory uses the default filter (`objectclass=*`).

Example 3:
The following LDAP URL retrieves the `cn`, `mail`, and `telephoneNumber` attributes of the entry for Barbara Jensen:

```
ldap://ldap.example.com/cn=Barbara%20Jensen,dc=example,dc=com?c
n,mail,telephoneNumber
```

- Because no search scope is specified, the search is restricted to the base entry `cn=Barbara Jensen,dc=example,dc=com`.
- Because no filter is specified, the directory uses the default filter `(objectclass=*).`

Example 4:
The following LDAP URL specifies a search for entries that have the surname `Jensen` and are at any level under `dc=example,dc=com`:

```
ldap://ldap.example.com/dc=example,dc=com??sub?(sn=Jensen)
```

- Because no attributes are specified, the search returns all attributes.
- Because the search scope is `sub`, the search encompasses the base entry `dc=example,dc=com` and entries at all levels under the base entry.

Example 5:

The following LDAP URL specifies a search for the object class for all entries one level under `dc=example,dc=com`:

```
ldap://ldap.example.com/dc=example,dc=com?objectClass?one
```

- Because the search scope is `one`, the search encompasses all entries one level under the base entry `dc=example,dc=com`. The search scope does not include the base entry.
- Because no filter is specified,the directory uses the default filter (`objectclass=*`).

NOTE The syntax for LDAP URLs does not include any means for specifying credentials or passwords. Search requests initiated through LDAP URLs are unauthenticated, unless the LDAP client that supports LDAP URLs provides an authentication mechanism. For example, Directory Server Gateway supports authentication.

Appendix D

Internationalization

Red Hat Directory Server (Directory Server) allows you to store, manage, and search for entries and their associated attributes in a number of different languages. An internationalized directory can be an invaluable corporate resource, providing employees and business partners with immediate access to the information they need in the languages they can understand.

The directory supports all international characters set by default because directory data is stored in UTF-8. Further, Directory Server allows you to specify matching rules and collation orders based on language preferences in search operations.

NOTE You must use ASCII characters for attribute and object class names.

This appendix contains the following sections:

About Locales

Directory Server provides support for multiple languages through the use of locales. A locale identifies language-specific information about how users of a specific region, culture, or custom expect data to be presented, including how data of a given language is interpreted and how data is to be sorted, or *collated*.

In addition, the locale information indicates what code page should be used to represent a given language. A code page is an internal table that the operating system uses to relate keyboard keys to character font screen displays.

More specifically, a locale specifies:

- Collation order — The collation order provides language and cultural-specific information about how the characters of a given language are to be sorted. It identifies things like the sequence of the letters in the alphabet, how to compare letters with accents to letters without accents, and if there are any characters that can be ignored when comparing strings. The collation order also takes into account culture-specific information about a language, such as the direction in which the language is read (left to right, right to left, or up and down).
- Character type — The character type distinguishes alphabetic characters from numeric or other characters. In addition, it defines the mapping of upper-case to lower-case letters. For example, in some languages, the pipe (|) character is considered punctuation while in others it is considered alphabetic.
- Monetary format — The monetary format specifies the monetary symbol used by a specific region, whether the symbol goes before or after its value, and how monetary units are represented.
- Time/date format — The time and date format indicates the customary formatting for times and dates in the region. The time and date format indicates whether dates are customarily represented in the `mm/dd/yy` (month, day, year) or `dd/mm/yy` (day, month, year) format and specifies what the days of the week and month are in a given language. For example, the date January 10, 1996, is represented as 10.leden 1996 in Czechoslovakian and 10 janvier 1996 in French.

Because a locale describes cultural, customary, and regional differences in addition to mechanical language differences, the directory data can both be translated into the specific languages understood by your users as well as be presented in a way that users in a given region expect.

Locale information is automatically copied to this directory during Directory Server installation:

```
serverRoot/lib/nls/locale30
```

Identifying Supported Locales

When performing directory operations that require you to specify a locale, such as a search operation, you can use a language tag or a collation order object identifier (OID).

A language tag is a string that begins with the two-character lowercase language code that identifies the language (as defined in ISO Standard 639). If necessary to distinguish regional differences in language, the language tag may also contain a country code, which is a two-character string (as defined in ISO Standard 3166). The language code and country code are separated by a hyphen. For example, the language tag used to identify the British English locale is en-GB.

An object identifier (OID) is a decimal number used to uniquely identify an object, such as an attribute or object class. The OIDs you use when searching or indexing an internationalized directory identify specific 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.

When performing an international search in the directory, use either the language tag or the OID to identify the collation order you want to use. However, when setting up an international index, you must use the OIDs. For more information on indexing, see chapter 10, “Managing Indexes.”

Table D-1 lists each locale supported by Directory Server and identifies the associated language tags and OIDs.

Table D-1 Supported Locales

Locale	**Language Tag**	**Collation Order Object Identifiers (OIDs)**
Albanian	`sq`	`2.16.840.1.113730.3.3.2.44.1`
Arabic	`ar`	`2.16.840.1.113730.3.3.2.1.1`
Byelorussian	`be`	`2.16.840.1.113730.3.3.2.2.1`
Bulgarian	`bg`	`2.16.840.1.113730.3.3.2.3.1`
Catalan	`ca`	`2.16.840.1.113730.3.3.2.4.1`
Chinese (Simplified)	`zh`	`2.16.840.1.113730.3.3.2.49.1`
Chinese (Traditional)	`zh-TW`	`2.16.840.1.113730.3.3.2.50.1`
Croatian	`hr`	`2.16.840.1.113730.3.3.2.22.1`
Czechoslovakian	`cs`	`2.16.840.1.113730.3.3.2.5.1`
Danish	`da`	`2.16.840.1.113730.3.3.2.6.1`

Table D-1 Supported Locales *(Continued)*

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
English (US)	`en` or `en-US`	`2.16.840.1.113730.3.3.2.11.1`
Estonian	`et`	`2.16.840.1.113730.3.3.2.16.1`
Finnish	`fi`	`2.16.840.1.113730.3.3.2.17.1`
French	`fr` or `fr-FR`	`2.16.840.1.113730.3.3.2.18.1`
German	`de`	`2.16.840.1.113730.3.3.2.7.1`
Greek	`el`	`2.16.840.1.113730.3.3.2.10.1`
Hebrew	`iw`	`2.16.840.1.113730.3.3.2.27.1`
Hungarian	`hu`	`2.16.840.1.113730.3.3.2.23.1`
Icelandic	`is`	`2.16.840.1.113730.3.3.2.24.1`
Japanese	`ja`	`2.16.840.1.113730.3.3.2.28.1`
Korean	`ko`	`2.16.840.1.113730.3.3.2.29.1`
Latvian, Lettish	`lv`	`2.16.840.1.113730.3.3.2.31.1`
Lithuanian	`lt`	`2.16.840.1.113730.3.3.2.30.1`
Macedonian	`mk`	`2.16.840.1.113730.3.3.2.32.1`
Norwegian	`no`	`2.16.840.1.113730.3.3.2.35.1`
Polish	`pl`	`2.16.840.1.113730.3.3.2.38.1`
Romanian	`ro`	`2.16.840.1.113730.3.3.2.39.1`
Russian	`ru`	`2.16.840.1.113730.3.3.2.40.1`
Serbian (Cyrillic)	`sr`	`2.16.840.1.113730.3.3.2.45.1`
Serbian (Latin)	`sh`	`2.16.840.1.113730.3.3.2.41.1`
Slovakian	`sk`	`2.16.840.1.113730.3.3.2.42.1`
Slovenian	`sl`	`2.16.840.1.113730.3.3.2.43.1`
Spanish	`es` or `es-ES`	`2.16.840.1.113730.3.3.2.15.1`
Swedish	`sv`	`2.16.840.1.113730.3.3.2.46.1`
Turkish	`tr`	`2.16.840.1.113730.3.3.2.47.1`
Ukranian	`uk`	`2.16.840.1.113730.3.3.2.48.1`

Supported Language Subtypes

Language subtypes can be used by clients to determine specific values for which to search. For more information on using language subtypes, see "Adding an Attribute Subtype," on page 55.

Table D-2 contains the list of supported language subtypes.

Table D-2 Supported Language Subtypes

Language tag	Language
af	Afrikaans
be	Byelorussian
bg	Bulgarian
ca	Catalan
cs	Czechoslovakian
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fi	Finnish
fo	Faroese
fr	French
ga	Irish
gl	Galician
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ko	Korean

Table D-2 Supported Language Subtypes *(Continued)*

Language tag	Language
`nl`	Dutch
`no`	Norwegian
`pl`	Polish
`pt`	Portuguese
`ro`	Romanian
`ru`	Russian
`sk`	Slovakian
`sl`	Slovenian
`sq`	Albanian
`sr`	Serbian
`sv`	Swedish
`tr`	Turkish
`uk`	Ukrainian
`zh`	Chinese

Troubleshooting Matching Rules

International collation order matching rules may not behave consistently. Some forms of matching-rule invocation do not work correctly, producing incorrect search results. For example, the following rules do not work:

```
./ldapsearch -p 9001 -D
"uid=gfarmer,ou=people,dc=example,dc=com" -w ruling -b
"dc=example,dc=com" "sn:2.16.840.1.113730.3.3.2.7.1:==passin"
```

```
./ldapsearch -p 9001 -D
"uid=gfarmer,ou=people,dc=example,dc=com" -w ruling -b
"dc=example,dc=com" "sn:de:==passin"
```

However, the rules listed below will work (note the `.3` in bold):

```
./ldapsearch -p 9001 -D
"uid=gfarmer,ou=people,dc=example,dc=com" -w ruling -b
"dc=example,dc=com" "sn:2.16.840.1.113730.3.3.2.7.1.3:=passin"
```

```
./ldapsearch -p 9001 -D
"uid=gfarmer,ou=people,dc=example,dc=com" -w ruling -b
"dc=example,dc=com" "sn:de.3:=passin"
```

Glossary

access control instruction *See ACI.*

ACI *Also Access Control Instruction.* An instruction that grants or denies permissions to entries in the directory.

access control list *See ACL.*

ACL *Also Access Control List.* The mechanism for controlling access to your directory.

access rights In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

account inactivation Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

All IDs Threshold A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

All IDs token A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index Allows for efficient approximate or “sounds-like” searches.

attribute Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list A list of required and optional attributes for a given entry type or object class.

authenticating directory server In pass-through authentication (PTA), the authenticating Directory Server is the Directory Server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.

authentication (1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

authentication certificate Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

base DN Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.

base distinguished name *See base DN.*

bind DN Distinguished name used to authenticate to Directory Server when performing an operation.

bind distinguished name *See bind DN.*

bind rule In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

branch entry An entry that represents the top of a subtree in the directory.

browser Software, such as Mozilla Firefox, used to request and view World Wide Web material stored as HTML files. The browser uses the HTTP protocol to communicate with the host server.

browsing index *Also virtual view index*. Speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branchpoint in the directory tree to improve display performance.

CA *See Certificate Authority*.

cascading replication In a cascading replication scenario, one server, often called the hub supplier, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. It receives updates from the supplier server that holds the master copy of the data and in turn supplies those updates to the consumer.

certificate A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in the directory as user object attributes.

Certificate Authority Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. *Also known as a CA.*

CGI *Common Gateway Interface*. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts and can be written in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.

chaining A method for relaying requests to another server. Results for the request are collected, compiled, and then returned to the client.

changelog A changelog is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers or on other masters, in the case of multi-master replication.

character type Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

CIR *See consumer-initiated replication*.

class definition Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.

class of service *See CoS*.

classic CoS A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.

client *See LDAP client*.

code page An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.

collation order Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

consumer Server containing replicated directory trees or subtrees from a supplier server.

consumer-initiated replication Replication configuration where consumer servers pull directory data from supplier servers.

consumer server In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.

CoS A method for sharing attributes between entries in a way that is invisible to applications.

CoS definition entry Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry Contains a list of the shared attribute values. *Also template entry.*

daemon A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.

DAP *Directory Access Protocol.* The ISO X.500 standard protocol that provides client access to the directory.

data master The server that is the master source of a particular piece of data.

database link An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.

default index One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.

definition entry *See CoS definition entry*.

Directory Access Protocol *See DAP.*

directory tree The logical representation of the information stored in the directory. It mirrors the tree model used by most filesystems, with the tree's root point appearing at the top of the hierarchy. Also known as DIT.

Directory Manager The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the Directory Manager.

Directory Server Gateway *Also DSGW.* A collection of CGI forms that allows a browser to perform LDAP client functions, such as querying and accessing a Directory Server, from a web browser.

directory service A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

distinguished name String representation of an entry's name and location in an LDAP directory.

DIT *See directory tree.*

DN *see distinguished name.*

DM *See Directory Manager.*

DNS *Domain Name System.* The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as `www.example.com`). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.

DNS alias A DNS alias is a hostname that the DNS server knows points to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as `www.`*yourdomain*`.`*domain* might point to a real machine called `realthing.`*yourdomain*`.`*domain* where the server currently exists.

DSGW *See Directory Server Gateway.*

entry A group of lines in the LDIF file that contains information about an object.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

file extension The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename `index.html` the file extension is `html`.

file type The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter A constraint applied to a directory query that restricts the information returned.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

gateway *See Directory Server Gateway*.

general access When granted, indicates that all authenticated users can access directory information.

GSS-API *Generic Security Services*. The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

hostname A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, `www.example.com` is the machine `www` in the subdomain `example` and `com` domain.

HTML *Hypertext Markup Language.* The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Mozilla Firefox how to display text, position graphics, and form items and to display links to other pages.

HTTP *Hypertext Transfer Protocol.* The method for exchanging information between HTTP servers and clients.

HTTPD An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.

HTTP-NG The next generation of Hypertext Transfer Protocol.

HTTPS A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.

hub supplier In the context of replication, a server that holds a replica that is copied from a different server, and, in turn, replicates it to a third server. *See also cascading replication.*

index key Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index Speeds up searches for information in international directories.

International Standards Organization *See ISO.*

IP address *Also Internet Protocol address.* A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10).

ISO International Standards Organization.

knowledge reference Pointers to directory information stored in different databases.

LDAP *Lightweight Directory Access Protocol.* Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAPv3 Version 3 of the LDAP protocol, upon which Directory Server bases its schema format.

LDAP client Software used to request and view LDAP entries from an LDAP Directory Server. *See also browser*.

LDAP Data Interchange Format *See LDAP Data Interchange Format*.

LDAP URL Provides the means of locating Directory Servers using DNS and then completing the query via LDAP. A sample LDAP URL is `ldap://ldap.example.com`.

LDBM database A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.

LDIF *LDAP Data Interchange Format.* Format used to represent Directory Server entries in text form.

leaf entry An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol *See LDAP*.

locale Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

managed object A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role Allows creation of an explicit enumerated list of members.

management information base *See MIB*.

mapping tree A data structure that associates the names of suffixes (subtrees) with databases.

master agent *See SNMP master agent*.

matching rule Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5 A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data that is unique with high probability and is mathematically extremely hard to produce; a piece of data that will produce the same message digest.

MD5 signature A message digest produced by the MD5 algorithm.

MIB *Management Information Base.* All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree-like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace *Management Information Base namespace.* The means for directory data to be named and referenced. *Also called the directory tree.*

monetary format Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.

multi-master replication An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a changelog for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor The server containing the database link that communicates with the remote server.

n + 1 directory problem The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions Multiple entries with the same distinguished name.

nested role Allows the creation of roles that contain other roles.

network management application Network Management Station component that graphically displays information about SNMP managed devices (which device is up or down, which and how many error messages were received, etc.).

network management station *See NMS*.

NIS *Network Information Service.* A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, filesystems, and network parameters throughout a network of computers.

NMS *Also Network Management Station.* Powerful workstation with one or more network management applications installed.

ns-slapd Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server. *See also slapd*.

object class Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier *Also OID.* A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

OID *See object identifier*.

operational attribute Contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

parent access When granted, indicates that users have access to entries below their own in the directory tree if the bind DN is the parent of the targeted entry.

pass-through authentication *See PTA.*

pass-through subtree In pass-through authentication, the PTA directory server will pass through bind requests to the authenticating directory server from all clients whose DN is contained in this subtree.

password file A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as `/etc/passwd` because of where it is kept.

password policy A set of rules that governs how passwords are used in a given directory.

permission In the context of access control, permission states whether access to the directory information is granted or denied and the level of access that is granted or denied. *See access rights.*

PDU *Also Protocol Data Unit.* Encoded messages which form the basis of data exchanges between SNMP devices.

pointer CoS A pointer CoS identifies the template entry using the template DN only.

presence index Allows searches for entries that contain a specific indexed attribute.

protocol A set of rules that describes how devices on a network exchange information.

protocol data unit *See PDU.*

proxy authentication A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.

proxy DN Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.

PTA *Also Pass-through authentication.* Mechanism by which one Directory Server consults another to check bind credentials.

PTA directory server In pass-through authentication (PTA), the PTA Directory Server is the server that sends (passes through) bind requests it receives to the authenticating directory server.

PTA LDAP URL In pass-through authentication, the URL that defines the authenticating directory server, pass-through subtree(s), and optional parameters.

RAM *Random access memory.* The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.

rc.local A file on Unix machines that describes programs that are run when the machine starts. It is also called `/etc/rc.local` because of its location.

RDN *Also Relative Distinguished Name.* The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name.

referential integrity Mechanism that ensures that relationships between related entries are maintained within the directory.

referral (1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.

(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.

replica A database that participates in replication.

read-only replica A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.

read-write replica A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.

relative distinguished name *See RDN.*

replication Act of copying directory trees or subtrees from supplier servers to consumer servers.

replication agreement Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the consumer servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

RFC *Request for Comments*. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

role An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

root suffix The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

SASL *Also Simple Authentication and Security Layer*. An authentication framework for clients as they attempt to bind to a directory.

schema Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

schema checking Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default, and users will receive an error if they try to save an entry that does not conform to the schema.

Secure Sockets Layer *See SSL.*

self access When granted, indicates that users have access to their own entries if the bind DN matches the targeted entry.

Server Console Java-based application that allows you to perform administrative management of your Directory Server from a GUI.

server daemon The server daemon is a process that, once running, listens for and accepts requests from clients.

server service A process on Windows that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.

server root A directory on the server machine dedicated to holding the server program and configuration, maintenance, and information files.

Server Selector Interface that allows you select and configure servers using a browser.

service A background process on a Windows machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE *Server Instance Entry*. The ID assigned to an instance of Directory Server during installation.

Simple Authentication and Security Layer *See SASL.*

Simple Network Management Protocol *See SNMP.*

single-master replication The most basic replication scenario in which two servers each hold a copy of the same read-write replicas to consumer servers. In a single-master replication scenario, the supplier server maintains a changelog.

SIR *See supplier-initiated replication.*

slapd LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication. *See also ns-slapd.*

SNMP *Also Simple Network Management Protocol.* Used to monitor and manage application processes running on the servers by exchanging data about network activity.

SNMP master agent Software that exchanges information between the various subagents and the NMS.

SNMP subagent Software that gathers information about the managed device and passes the information to the master agent. *Also subagent.*

SSL *Also Secure Sockets Layer*. A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP.

standard index index maintained by default.

sub suffix A branch underneath a root suffix.

subagent *See SNMP subagent.*

substring index Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.

suffix The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

superuser The most privileged user available on Unix machines. The superuser has complete access privileges to all files on the machine. *Also called root.*

supplier Server containing the master copy of directory trees or subtrees that are replicated to consumer servers.

supplier server In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication Replication configuration where supplier servers replicate directory data to consumer servers.

symmetric encryption Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index Cannot be deleted or modified as it is essential to Directory Server operations.

target In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry The entries within the scope of a CoS.

TCP/IP *Transmission Control Protocol/Internet Protocol.* The main network protocol for the Internet and for enterprise (company) networks.

template entry *See CoS template entry*.

time / date format Indicates the customary formatting for times and dates in a specific region.

TLS *Also Transport Layer Security.* The new standard for secure socket layers; a public key based protocol.

topology The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security *See TLS*.

uid A unique number associated with each user on a Unix system.

URL *Uniform Resource Locator.* The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is *protocol*://*machine*:*port*/*document*. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

virtual list view index *Also browsing index*. Speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branchpoint in the directory tree to improve display performance.

X.500 standard The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation.

Index