



# Red Hat

## Red Hat Directory Server 11

### Administration Guide

Basic and advanced administration of Directory Server



# Red Hat Directory Server 11 Administration Guide

---

Basic and advanced administration of Directory Server

Marc Muehlfeld  
Red Hat Customer Content Services  
[mmuehlfeld@redhat.com](mailto:mmuehlfeld@redhat.com)

Petr Bokoč  
Red Hat Customer Content Services

Tomáš Čapek  
Red Hat Customer Content Services

Petr Kovář  
Red Hat Customer Content Services

Ella Deon Ballard  
Red Hat Customer Content Services

## Legal Notice

Copyright © 2021 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This guide covers both GUI and command-line procedures for managing Directory Server instances and databases.

## Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .....	6
<b>CHAPTER 1. GENERAL DIRECTORY SERVER MANAGEMENT TASKS .....</b>	<b>7</b>
1.1. SYSTEM REQUIREMENTS .....	7
1.2. FILE LOCATIONS .....	7
1.3. SUPPORTED METHODS TO CONFIGURE DIRECTORY SERVER .....	7
1.4. LOGGING INTO DIRECTORY SERVER USING THE WEB CONSOLE .....	7
1.5. STARTING AND STOPPING A DIRECTORY SERVER INSTANCE .....	8
1.6. CREATING A NEW DIRECTORY SERVER INSTANCE .....	9
1.7. REMOVING A DIRECTORY SERVER INSTANCE .....	9
1.8. SETTING DIRECTORY SERVER CONFIGURATION PARAMETERS .....	10
1.9. CHANGING THE LDAP AND LDAPS PORT NUMBERS .....	12
1.10. USING DIRECTORY SERVER PLUG-INS .....	13
<b>CHAPTER 2. CONFIGURING DIRECTORY DATABASES .....</b>	<b>18</b>
2.1. CREATING AND MAINTAINING SUFFIXES .....	18
2.2. CREATING AND MAINTAINING DATABASES .....	24
2.3. CREATING AND MAINTAINING DATABASE LINKS .....	31
2.4. CONFIGURING CASCADING CHAINING .....	44
2.5. USING REFERRALS .....	50
2.6. VERIFYING THE INTEGRITY OF BACK-END DATABASES .....	53
<b>CHAPTER 3. MANAGING DIRECTORY ENTRIES .....</b>	<b>55</b>
3.1. PROVIDING INPUT TO THE LDAPADD, LDAPMODIFY, AND LDAPDELETE UTILITIES .....	55
3.2. THE CONTINUOUS OPERATION MODE .....	57
3.3. ADDING AN ENTRY .....	57
3.4. UPDATING A DIRECTORY ENTRY .....	58
3.5. DELETING AN ENTRY .....	60
3.6. RENAMING AND MOVING AN ENTRY .....	61
3.7. USING SPECIAL CHARACTERS .....	65
3.8. USING BINARY ATTRIBUTES .....	65
3.9. UPDATING AN ENTRY IN AN INTERNATIONALIZED DIRECTORY .....	65
<b>CHAPTER 4. TRACKING MODIFICATIONS TO DIRECTORY ENTRIES .....</b>	<b>67</b>
4.1. TRACKING MODIFICATIONS TO THE DATABASE THROUGH UPDATE SEQUENCE NUMBERS .....	67
4.2. TRACKING ENTRY MODIFICATIONS THROUGH OPERATIONAL ATTRIBUTES .....	71
4.3. TRACKING THE BIND DN FOR PLUG-IN INITIATED UPDATES .....	72
4.4. TRACKING PASSWORD CHANGE TIMES .....	74
<b>CHAPTER 5. MAINTAINING REFERENTIAL INTEGRITY .....</b>	<b>75</b>
5.1. HOW REFERENTIAL INTEGRITY WORKS .....	75
5.2. USING REFERENTIAL INTEGRITY WITH REPLICATION .....	76
5.3. ENABLING REFERENTIAL INTEGRITY .....	76
5.4. THE REFERENTIAL INTEGRITY UPDATE INTERVAL .....	77
5.5. DISPLAYING AND MODIFYING THE ATTRIBUTE LIST .....	79
5.6. CONFIGURING SCOPE FOR THE REFERENTIAL INTEGRITY .....	80
<b>CHAPTER 6. POPULATING DIRECTORY DATABASES .....</b>	<b>84</b>
6.1. IMPORTING DATA .....	84
6.2. EXPORTING DATA .....	89
6.3. BACKING UP DIRECTORY SERVER .....	95
6.4. RESTORING DIRECTORY SERVER .....	100

<b>CHAPTER 7. MANAGING ATTRIBUTES AND VALUES</b>	105
7.1. ENFORCING ATTRIBUTE UNIQUENESS	105
7.2. ASSIGNING CLASS OF SERVICE	108
7.3. LINKING ATTRIBUTES TO MANAGE ATTRIBUTE VALUES	121
7.4. ASSIGNING AND MANAGING UNIQUE NUMERIC ATTRIBUTE VALUES	125
<b>CHAPTER 8. ORGANIZING AND GROUPING ENTRIES</b>	132
8.1. USING GROUPS	132
8.2. USING ROLES	152
8.3. AUTOMATICALLY CREATING DUAL ENTRIES	157
8.4. USING VIEWS	164
8.5. MANAGING ORGANIZATIONAL UNITS	166
<b>CHAPTER 9. CONFIGURING SECURE CONNECTIONS</b>	168
9.1. REQUIRING SECURE CONNECTIONS	168
9.2. SETTING A MINIMUM STRENGTH FACTOR	168
9.3. MANAGING THE NSS DATABASE USED BY DIRECTORY SERVER	169
9.4. ENABLING TLS	179
9.5. DISPLAYING THE ENCRYPTION PROTOCOLS ENABLED IN DIRECTORY SERVER	186
9.6. SETTING THE MINIMUM TLS ENCRYPTION PROTOCOL VERSION	187
9.7. SETTING THE HIGHEST TLS ENCRYPTION PROTOCOL VERSION	187
9.8. USING HARDWARE SECURITY MODULES	187
9.9. USING CERTIFICATE-BASED CLIENT AUTHENTICATION	188
9.10. SETTING UP SASL IDENTITY MAPPING	191
9.11. USING KERBEROS GSS-API WITH SASL	197
9.12. SETTING SASL MECHANISMS	199
9.13. USING SASL WITH LDAP CLIENTS	199
<b>CHAPTER 10. CONFIGURING ATTRIBUTE ENCRYPTION</b>	201
10.1. ENCRYPTION KEYS	201
10.2. ENCRYPTION CIPHERS	202
10.3. CONFIGURING ATTRIBUTE ENCRYPTION	202
10.4. EXPORTING AND IMPORTING AN ENCRYPTED DATABASE	205
10.5. UPDATING THE TLS CERTIFICATES USED FOR ATTRIBUTE ENCRYPTION	206
<b>CHAPTER 11. MANAGING FIPS MODE SUPPORT</b>	208
Enabling FIPS Mode Support	208
Disabling FIPS Mode Support	208
<b>CHAPTER 12. MANAGING THE DIRECTORY SCHEMA</b>	209
12.1. OVERVIEW OF SCHEMA	209
12.2. MANAGING OBJECT IDENTIFIERS	214
12.3. CREATING AN OBJECT CLASS	214
12.4. UPDATING AN OBJECT CLASS	215
12.5. REMOVING AN OBJECT CLASS	216
12.6. CREATING AN ATTRIBUTE	217
12.7. UPDATING AN ATTRIBUTE	218
12.8. REMOVING AN ATTRIBUTE	220
12.9. CREATING CUSTOM SCHEMA FILES	221
12.10. DYNAMICALLY RELOADING SCHEMA	223
12.11. TURNING SCHEMA CHECKING ON AND OFF	225
12.12. USING SYNTAX VALIDATION	226
<b>CHAPTER 13. MANAGING INDEXES</b>	231
13.1. ABOUT INDEXES	231

13.2. CREATING STANDARD INDEXES	235
13.3. CREATING NEW INDEXES TO EXISTING DATABASES	238
13.4. CREATING BROWSING INDEXES	239
13.5. CHANGING THE INDEX SORT ORDER	243
13.6. CHANGING THE WIDTH FOR INDEXED SUBSTRING SEARCHES	243
13.7. DELETING INDEXES	244
<b>CHAPTER 14. FINDING DIRECTORY ENTRIES .....</b>	<b>247</b>
14.1. USING LDAPSEARCH	247
14.2. LDAP SEARCH FILTERS	250
14.3. EXAMPLES OF COMMON LDAPSEARCHES	264
14.4. IMPROVING SEARCH PERFORMANCE THROUGH RESOURCE LIMITS	268
14.5. USING PERSISTENT SEARCH	273
14.6. SEARCHING WITH SPECIFIED CONTROLS	274
<b>CHAPTER 15. MANAGING REPLICATION .....</b>	<b>280</b>
15.1. REPLICATION OVERVIEW	280
15.2. CONFIGURING BOOTSTRAP CREDENTIALS	283
15.3. SINGLE-SUPPLIER REPLICATION	284
15.4. MULTI-SUPPLIER REPLICATION	290
15.5. CASCADING REPLICATION	301
15.6. CONFIGURING REPLICATION PARTNERS TO USE CERTIFICATE-BASED AUTHENTICATION	311
15.7. PROMOTING A CONSUMER OR HUB TO A SUPPLIER	313
15.8. ABOUT INITIALIZING A CONSUMER	314
15.9. DISABLING AND RE-ENABLING REPLICATION	316
15.10. REMOVING A DIRECTORY SERVER INSTANCE FROM THE REPLICATION TOPOLOGY	316
15.11. MANAGING ATTRIBUTES WITHIN FRACTIONAL REPLICATION	319
15.12. MANAGING DELETED ENTRIES WITH REPLICATION	322
15.13. CONFIGURING CHANGELOG ENCRYPTION	323
15.14. REMOVING THE CHANGELOG	324
15.15. EXPORTING THE REPLICATION CHANGELOG	325
15.16. IMPORTING THE REPLICATION CHANGELOG FROM AN LDIF-FORMATTED CHANGELOG DUMP	325
15.17. MOVING THE REPLICATION CHANGELOG DIRECTORY	325
15.18. TRIMMING THE REPLICATION CHANGELOG	327
15.19. FORCING REPLICATION UPDATES	329
15.20. SETTING REPLICATION TIMEOUT PERIODS	330
15.21. USING THE RETRO CHANGELOG PLUG-IN	330
15.22. DISPLAYING THE STATUS OF A SPECIFIC REPLICATION AGREEMENT	332
15.23. MONITORING THE REPLICATION TOPOLOGY	333
15.24. COMPARING TWO DIRECTORY SERVER INSTANCES	335
15.25. SOLVING COMMON REPLICATION CONFLICTS	337
15.26. TROUBLESHOOTING REPLICATION-RELATED PROBLEMS	341
<b>CHAPTER 16. SYNCHRONIZING RED HAT DIRECTORY SERVER WITH MICROSOFT ACTIVE DIRECTORY ...</b>	<b>344</b>
16.1. ABOUT WINDOWS SYNCHRONIZATION	344
16.2. SUPPORTED ACTIVE DIRECTORY VERSIONS	347
16.3. SYNCHRONIZING PASSWORDS	347
16.4. SETTING UP SYNCHRONIZATION BETWEEN ACTIVE DIRECTORY AND DIRECTORY SERVER	348
16.5. SYNCHRONIZING USERS	358
16.6. SYNCHRONIZING GROUPS	363
16.7. CONFIGURING UNI-DIRECTIONAL SYNCHRONIZATION	367
16.8. CONFIGURING MULTIPLE SUBTREES AND FILTERS IN WINDOWS SYNCHRONIZATION	368
16.9. SYNCHRONIZING POSIX ATTRIBUTES FOR USERS AND GROUPS	369

16.10. DELETING AND RESURRECTING ENTRIES	370
16.11. SENDING SYNCHRONIZATION UPDATES	371
16.12. TROUBLESHOOTING	376
<b>CHAPTER 17. SETTING UP CONTENT SYNCHRONIZATION USING THE SYNCREPL PROTOCOL .....</b>	<b>378</b>
17.1. CONFIGURING THE CONTENT SYNCHRONIZATION PLUG-IN USING THE COMMAND LINE	378
<b>CHAPTER 18. MANAGING ACCESS CONTROL .....</b>	<b>380</b>
18.1. ACCESS CONTROL PRINCIPLES	380
18.2. ACI PLACEMENT	380
18.3. ACI STRUCTURE	381
18.4. ACI EVALUATION	382
18.5. LIMITATIONS OF ACIS	382
18.6. HOW DIRECTORY SERVER HANDLES ACIS IN A REPLICATION TOPOLOGY	383
18.7. MANAGING ACIS	383
18.8. DEFINING TARGETS	384
18.9. DEFINING PERMISSIONS	393
18.10. DEFINING BIND RULES	395
18.11. CHECKING ACCESS RIGHTS ON ENTRIES (GET EFFECTIVE RIGHTS)	411
18.12. LOGGING ACCESS CONTROL INFORMATION	421
18.13. ADVANCED ACCESS CONTROL: USING MACRO ACIS	421
18.14. SETTING ACCESS CONTROLS ON DIRECTORY MANAGER	426
<b>CHAPTER 19. USING THE HEALTH CHECK FEATURE TO IDENTIFY PROBLEMS .....</b>	<b>428</b>
19.1. RUNNING THE DIRECTORY SERVER HEALTH CHECK	429
<b>CHAPTER 20. MANAGING USER AUTHENTICATION .....</b>	<b>432</b>
20.1. SETTING USER PASSWORDS	432
20.2. SETTING PASSWORD ADMINISTRATORS	432
20.3. CHANGING PASSWORDS STORED EXTERNALLY	433
20.4. MANAGING THE PASSWORD POLICY	434
20.5. CONFIGURING TEMPORARY PASSWORD RULES	439
20.6. UNDERSTANDING PASSWORD EXPIRATION CONTROLS	441
20.7. MANAGING THE DIRECTORY MANAGER PASSWORD	442
20.8. CHECKING ACCOUNT AVAILABILITY FOR PASSWORDLESS ACCESS	445
20.9. CONFIGURING A PASSWORD-BASED ACCOUNT LOCKOUT POLICY	447
20.10. CONFIGURING TIME-BASED ACCOUNT LOCKOUT POLICIES	449
20.11. REPLICATING ACCOUNT LOCKOUT ATTRIBUTES	455
20.12. ENABLING DIFFERENT TYPES OF BINDS	457
20.13. USING PASS-THROUGH AUTHENTICATION	462
20.14. USING ACTIVE DIRECTORY-FORMATTED USER NAMES FOR AUTHENTICATION	469
20.15. USING PAM FOR PASS-THROUGH AUTHENTICATION	471
20.16. MANUALLY INACTIVATING USERS AND ROLES	476
<b>CHAPTER 21. MONITORING SERVER AND DATABASE ACTIVITY .....</b>	<b>478</b>
21.1. TYPES OF DIRECTORY SERVER LOG FILES	478
21.2. DISPLAYING LOG FILES	478
21.3. CONFIGURING LOG FILES	479
21.4. GETTING ACCESS LOG STATISTICS	488
21.5. MONITORING THE LOCAL DISK FOR GRACEFUL SHUTDOWN	492
21.6. MONITORING SERVER ACTIVITY	492
21.7. MONITORING DATABASE ACTIVITY	492
21.8. MONITORING DATABASE LINK ACTIVITY	492
21.9. ENABLING AND DISABLING COUNTERS	492

---

21.10. MONITORING DIRECTORY SERVER USING SNMP	492
<b>CHAPTER 22. MAKING A HIGH-AVAILABILITY AND DISASTER RECOVERY PLAN .....</b>	<b>500</b>
22.1. IDENTIFYING POTENTIAL SCENARIOS	500
22.2. DEFINING THE TYPE OF ROLLOVER	501
22.3. IDENTIFYING USEFUL DIRECTORY SERVER FEATURES FOR DISASTER RECOVERY	501
22.4. DEFINING THE RECOVERY PROCESS	503
22.5. BASIC EXAMPLE: PERFORMING A RECOVERY	503
<b>CHAPTER 23. CREATING TEST ENTRIES .....</b>	<b>505</b>
23.1. CREATING AN LDIF FILE WITH EXAMPLE USER ENTRIES	505
23.2. CREATING AN LDIF FILE WITH EXAMPLE GROUP ENTRIES	506
23.3. CREATING AN LDIF FILE WITH AN EXAMPLE COS DEFINITION	506
23.4. CREATING AN LDIF FILE WITH EXAMPLE MODIFICATION STATEMENTS	507
23.5. CREATING AN LDIF FILE WITH NESTED EXAMPLE ENTRIES	507
<b>APPENDIX A. USING LDAP CLIENT TOOLS .....</b>	<b>508</b>
A.1. RUNNING EXTENDED OPERATIONS	508
A.2. COMPARING ENTRIES	509
A.3. CHANGING PASSWORDS	510
A.4. GENERATING LDAP URLs	511
<b>APPENDIX B. LDAP DATA INTERCHANGE FORMAT .....</b>	<b>514</b>
B.1. ABOUT THE LDIF FILE FORMAT	514
B.2. CONTINUING LINES IN LDIF	515
B.3. REPRESENTING BINARY DATA	515
B.4. SPECIFYING DIRECTORY ENTRIES USING LDIF	517
B.5. DEFINING DIRECTORIES USING LDIF	521
B.6. STORING INFORMATION IN MULTIPLE LANGUAGES	523
<b>APPENDIX C. LDAP URLs .....</b>	<b>525</b>
C.1. COMPONENTS OF AN LDAP URL	525
C.2. ESCAPING UNSAFE CHARACTERS	526
C.3. EXAMPLES OF LDAP URLs	527
<b>APPENDIX D. INTERNATIONALIZATION .....</b>	<b>529</b>
D.1. ABOUT LOCALES	529
D.2. SUPPORTED LOCALES	529
D.3. SUPPORTED LANGUAGE SUBTYPES	530
D.4. SEARCHING AN INTERNATIONALIZED DIRECTORY	532
D.5. TROUBLESHOOTING MATCHING RULES	537
<b>APPENDIX E. REVISION HISTORY .....</b>	<b>538</b>

## MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see "[our CTO Chris Wright's message](#)".

# CHAPTER 1. GENERAL DIRECTORY SERVER MANAGEMENT TASKS

This chapter describes the general tasks of managing Directory Server instances.

## 1.1. SYSTEM REQUIREMENTS

See the corresponding section in the [Red Hat Directory Server 11 Release Notes](#).

## 1.2. FILE LOCATIONS

See the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

## 1.3. SUPPORTED METHODS TO CONFIGURE DIRECTORY SERVER

You can configure Directory Server using:

- the command-line utilities provided by Directory Server
- the web console
- LDAP Data Interchange Format (LDIF) statements sent to the server through an LDAP client



### IMPORTANT

The web console does not automatically display the latest settings if a user changes the configuration outside of the console's window. For example, if you change the configuration using the command line while the web console is open, the new settings are not automatically updated in the web console. This applies also if you change the configuration using the web console on a different computer. To work around the problem, manually refresh the web console in the browser if the configuration has been changed outside the console's window.

## 1.4. LOGGING INTO DIRECTORY SERVER USING THE WEB CONSOLE

The web console is a browser-based graphical user interface (GUI) that enables users to perform administrative tasks. The Directory Server package automatically installs the Directory Server user interface for the web console.

To open Directory Server in the web console:

1. Use a browser and connect to the web console running on port 9090 on the Directory Server host. For example:

```
https://server.example.com:9090
```

2. Log in as the **root** user or as a user with **sudo** privileges.
3. Select the **Red Hat Directory Server** entry.

The screenshot shows the Red Hat Directory Server Management interface. The top navigation bar includes 'Red Hat Directory Server Management' and a dropdown for 'slapd-instance\_name'. Below the navigation is a horizontal menu with 'Server Settings', 'Database', 'Replication', 'Schema', 'Plugins', and 'Monitoring'. On the left, a sidebar lists 'Overview', 'Logs', 'Networking', 'Accounts', 'Services', 'Diagnostic Reports', 'Kernel Dump', 'Red Hat Directory Server' (which is highlighted with a red box), and 'SELinux'. The main content area is titled 'Server Settings' and contains tabs for 'General Settings', 'Directory Manager', 'Disk Monitoring', and 'Advanced Settings'. Under 'General Settings', fields include 'Server Hostname' (server.example.com), 'LDAP Port' (389), 'LDAPS Port' (636), 'Listen Host Address' (empty), and 'Database Backup Directory' (/var/lib/dirsrv/slapd-instance). A sidebar on the right lists 'Tuning & Limits', 'Security', 'SASL Settings & Mappings', 'LDAP & Autobind', and 'Logging' (with sub-options for Access Log, Audit Log, Audit Failure Log, and Errors Log).

## 1.5. STARTING AND STOPPING A DIRECTORY SERVER INSTANCE

### 1.5.1. Starting and Stopping a Directory Server Instance Using the Command Line

Use the **dsctl** utility to start, stop, or restart an instance:

- To start the instance:

```
# dsctl instance_name start
```

- To stop the instance:

```
# dsctl instance_name stop
```

- To restart the instance:

```
# dsctl instance_name restart
```

Optionally, you can enable Directory Server instances to automatically start when the system boots:

- For a single instance:

```
# systemctl enable dirsrv@instance_name
```

- For all instances on a server:

```
# systemctl enable dirsrv.target
```

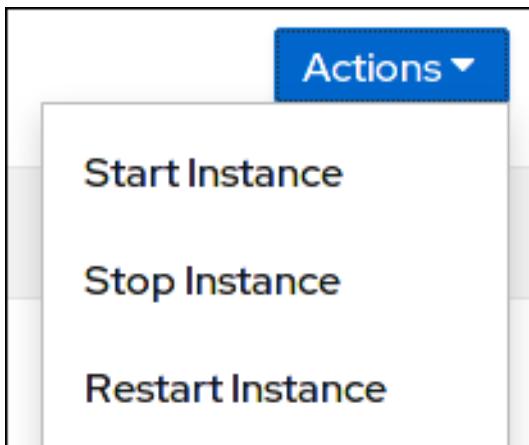
For further details, see the [Managing System Services](#) section in the *Red Hat System Administrator's Guide*.

### 1.5.2. Starting and Stopping a Directory Server Instance Using the Web Console

As an alternative to command line, you can use the web console to start, stop, or restart instances.

To start, stop, or restart a Directory Server instance:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Click the **Actions** button and select the action to execute:
  - **Start Instance**
  - **Stop Instance**
  - **Restart Instance**



## 1.6. CREATING A NEW DIRECTORY SERVER INSTANCE

For details, see the corresponding sections in the *Red Hat Directory Server Installation Guide*:

- [Setting up a new instance on the command line using a .inf file](#)
- [Setting up a new instance on the command line using the interactive installer](#)
- [Setting up a new instance using the web console](#)

## 1.7. REMOVING A DIRECTORY SERVER INSTANCE

If you run multiple instances on a server, you can remove individual instances of them.

When you remove an instance:

- The content of the **/var/lib/dirsrv/slapd-*instance\_name*** directory is removed.



### IMPORTANT

The **/var/lib/dirsrv/slapd-*instance\_name*** directory contains the database, as well as the backup and export directory. Before you remove an instance, backup this data.

- The **/etc/dirsrv/slapd-*instance\_name*** directory is renamed to **/etc/dirsrv/slapd-*instance\_name.removed***.

### 1.7.1. Removing an Instance Using the Command Line

To remove an instance using the command line:

```
# dsctl instance_name remove --do-it
Removing instance ...
Completed instance removal
```

## 1.7.2. Removing an Instance Using the Web Console

To remove an instance using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Click the **Actions** button, and select **Remove instance**.

## 1.8. SETTING DIRECTORY SERVER CONFIGURATION PARAMETERS

Directory Server stores its configuration in the **cn=config** directory entry. Each configuration parameter is an LDAP attribute and the parameter's value is the value set in this attribute.

### 1.8.1. Managing Configuration Parameters

You can set, update, and delete configuration parameters by:

- Using the **dsconf** utility:



#### NOTE

Red Hat recommends using the **dsconf** utility to manage the Directory Server configuration.

#### Example 1.1. Setting a Configuration Parameter Using **dsconf**

For example, to set the error log level to **16384**, update the **nsslapd-errorlog-level** parameter using the **dsconf** utility:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
errorlog-level=16384
```

For further details about using **dsconf**, see the **dsconf(8)** man page.

- Using the LDAP interface:

#### Example 1.2. Setting a Configuration Parameter using the LDAP Interface

For example, to set the error log level to **16384**, update the **nsslapd-errorlog-level** parameter using the LDAP interface:

```
# ldapmodify -D "cn=Directory Manager" -W -x -H ldap://server.example.com:389
```

```
dn: cn=config
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 16384
```

- Editing the `/etc/dirsrv/slappd-instance_name/dse.ldif` file.



### WARNING

As long as an instance starts successfully, do not manually edit this file because this can cause Directory Server to not work as expected, or the instance can fail to start.

## 1.8.2. Where Directory Server Stores its Configuration

Directory Server stores the configuration from the **cn=config** entry in the `/etc/dirsrv/slappd-instance_name/dse.ldif` file. The server stores only parameters you modified in this file. Attributes that are not listed, use their default value. This enables you to identify all configuration parameters you set in this instance by displaying the `/etc/dirsrv/slappd-instance_name/dse.ldif` file.



### IMPORTANT

Do not manually edit the `/etc/dirsrv/slappd-instance_name/dse.ldif` file as long as the instance starts successfully.

For details about how you can edit configuration parameters, see [Section 1.8.1, “Managing Configuration Parameters”](#).

## 1.8.3. Benefits of Using Default Values

If a parameter is not set, Directory Server uses the default value of this parameter. Using the default value has the benefit that new versions often provide optimized settings and increased security.

For example, if you do not set the **passwordStorageScheme** attribute, Directory Server automatically uses the strongest supported password storage scheme available. If a future update changes the default value to increase security, passwords will be automatically encrypted using the new storage scheme when a user sets a password.

### 1.8.3.1. Removing a Parameter to Use the Default Value

If a parameter is set and you want to use the default value instead, delete the parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config delete parameter_name
```

Note that you cannot delete certain parameters, such as **nsslapd-secureport** to reset them to their default. If you try to delete them, the server rejects the request with a **Server is unwilling to perform (53)** error.

## 1.9. CHANGING THE LDAP AND LDAPS PORT NUMBERS

By default, Directory Server uses port 389 for the LDAP and, if enabled, port 636 for the LDAPS protocol. You can change these port numbers, for example, to run multiple Directory Server instances on one host.



### IMPORTANT

The new ports you assign to the protocols for an instance must not be in use by any other service.

#### 1.9.1. Changing the Port Numbers Using the Command Line

To change the port numbers using the command line, update the following parameters:

- ***nsslapd-port***: Stores the port number the instance uses for the LDAP protocol.
- ***nsslapd-secureport***: Stores the port number the instance uses for the LDAPS protocol.

To change the port numbers of the LDAP and LDAPS protocol using the command line:

1. Optionally, display the currently configured port numbers for the instance:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-port  
nsslapd-secureport  
nsslapd-port: 389  
nsslapd-secureport: 636
```

2. To change the LDAP port:
  - a. Set the port for the LDAP protocol. For example, to set it to **1389**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-  
port=1389  
Successfully replaced "nsslapd-port"
```

3. To change the LDAPS port:
  - a. Set the port for the LDAPS protocol. For example, to set it to **1636**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-  
secureport=1636  
Successfully replaced "nsslapd-secureport"
```

4. Restart the instance:
  - b. Set the **ldap\_port\_t** type for the LDAPS port you assigned in the previous step:

```
# semanage port -a -t ldap_port_t -p tcp 1636
```

4. Restart the instance:

```
# dsctl instance_name restart
```

### 1.9.2. Changing the Port Numbers Using the Web Console

To change the port numbers of the LDAP and LDAPS protocol using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. To change the LDAP port:
  - a. Open the **Server Settings** menu.
  - b. On the **Server Settings** tab, fill the new port number into the **LDAP Port** field.
  - c. Click **Save**.
4. To change the LDAPS port:
  - a. Open the **Server Settings** menu.
  - b. On the **General Settings** tab, fill the new port number into the **LDAPS Port** field.
  - c. Click **Save**.
5. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

## 1.10. USING DIRECTORY SERVER PLUG-INS

Directory Server provides several core plug-ins, such as for replication, class of service, and attribute syntax validation. Core plug-ins are enabled by default.

Additionally, the Directory Server packages contain further plug-ins to enhance the functionality, such as for attribute uniqueness and attribute linking. However, not all of these plug-ins are enabled by default.

For further details, see the [Plug-in Implemented Server Functionality Reference](#) chapter in the *Red Hat Directory Server Configuration, Command, and File Reference*.

### 1.10.1. Listing Available Plug-ins

#### 1.10.1.1. Listing Available Plug-ins Using the Command Line

To list all available plug-ins using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin list
7-bit check
Account Policy Plugin
...
```

You require the exact name of the plug-in, for example, to enable or disable it using the command line.

### 1.10.1.2. Listing Available Plug-ins Using the Web Console

To display all available plug-ins using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.

Optionally, you can filter the plug-ins by entering a name into the **Filter Plugins** field.

### 1.10.2. Enabling Plug-ins Dynamically

By default, after enabling or disabling a plug-in, restart the instance to take effect.

If you set the **nsslapd-dynamic-plugins** parameter to **on**, you do not have to restart the instance after making the following changes:

- Enabling a plug-in
- Disabling a plug-in
- Changing the configuration of a plug-in

To enable this feature:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-dynamic-plugins=on
Successfully replaced "nsslapd-dynamic-plugins"
```

### 1.10.3. Enabling and Disabling Plug-ins

#### 1.10.3.1. Enabling and Disabling Plug-ins Using the Command Line

To enable or disable a plug-in using the command line, use the **dsconf** utility.



#### NOTE

The **dsconf** command requires that you provide the name of the plug-in. For details about displaying the names of all plug-ins, see [Section 1.10.1.1, “Listing Available Plug-ins Using the Command Line”](#).

For example, to enable the **Automember** plug-in:

1. Enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin automember enable
```

2. Restart the instance:

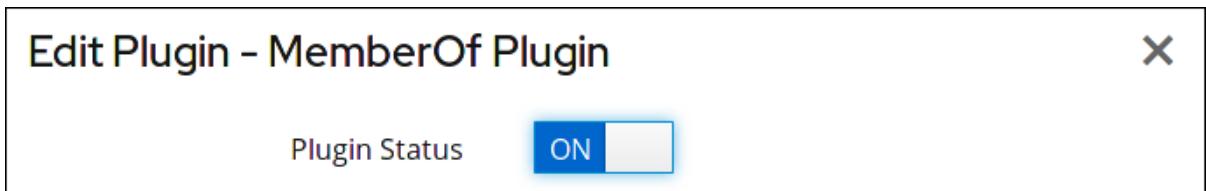
```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 1.10.3.2. Enabling and Disabling Plug-ins Using the Web Console

To enable or disable a plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **All Plugins** tab.
5. Click the **Edit Plugin** button to the right of the plug-in you want to enable or disable.
6. Change the status to **ON** to enable or to **OFF** to disable the plug-in.



#### Edit Plugin - MemberOf Plugin



Plugin Status

ON



7. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 1.10.4. Configuring Plug-ins

#### 1.10.4.1. Configuring Plug-ins Using the Command Line

To configure plug-in settings, use the **dsconf plugin** command:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin \
  plug-in-specific_subcommand ...
```

For a list of plug-ins you can configure, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin --help
```

#### 1.10.4.2. Configuring Plug-ins Using the Web Console

To configure a plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.

3. Select the **Plugins** menu.
4. Select the **All Plugins** tab.
5. Select the plug-in and click **Show Advanced Settings**.
6. Open the plug-in-specific tab.
7. Set the appropriate settings.
8. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 1.10.5. Setting the Plug-in Precedence

The plug-in precedence is the priority it has in the execution order of plug-ins. For pre- and post-operation plug-ins, this enables a plug-in to be executed and complete before the next plug-in is initiated, to let the next plug-in take advantage of the previous plug-in's results.

The precedence can be set to a value from **1** (highest priority) to **99** (lowest priority). If no precedence is set, the default is **50**.



#### WARNING

Set a precedence value only in custom plug-ins. Updating the value of core plug-ins can cause Directory Server to not work as expected and is not supported by Red Hat.

#### 1.10.5.1. Setting the Plug-in Precedence Using the Command Line

To update the precedence value of a plug-in using the command line:

1. Set precedence of the plug-in. For example, to set the precedence for the **example** plug-in to **1**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin edit example --precedence 1
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 1.10.5.2. Setting the Plug-in Precedence Using the Web Console

To update the precedence value of a plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select **All Plugins**.
5. Press the **Edit Plugin** button next to the plug-in for which you want to configure the precedence value.
6. Update the value in the **Plugin Precedence** field.
7. Click **Save**.
8. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

## CHAPTER 2. CONFIGURING DIRECTORY DATABASES

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

### 2.1. CREATING AND MAINTAINING SUFFIXES

Different pieces of the directory tree can be stored in different databases, and then these databases can be distributed across multiple servers. The directory tree contains branch points called *nodes*. These nodes may be associated with databases. A suffix is a node of the directory tree associated with a particular database. The following is a simple directory tree:

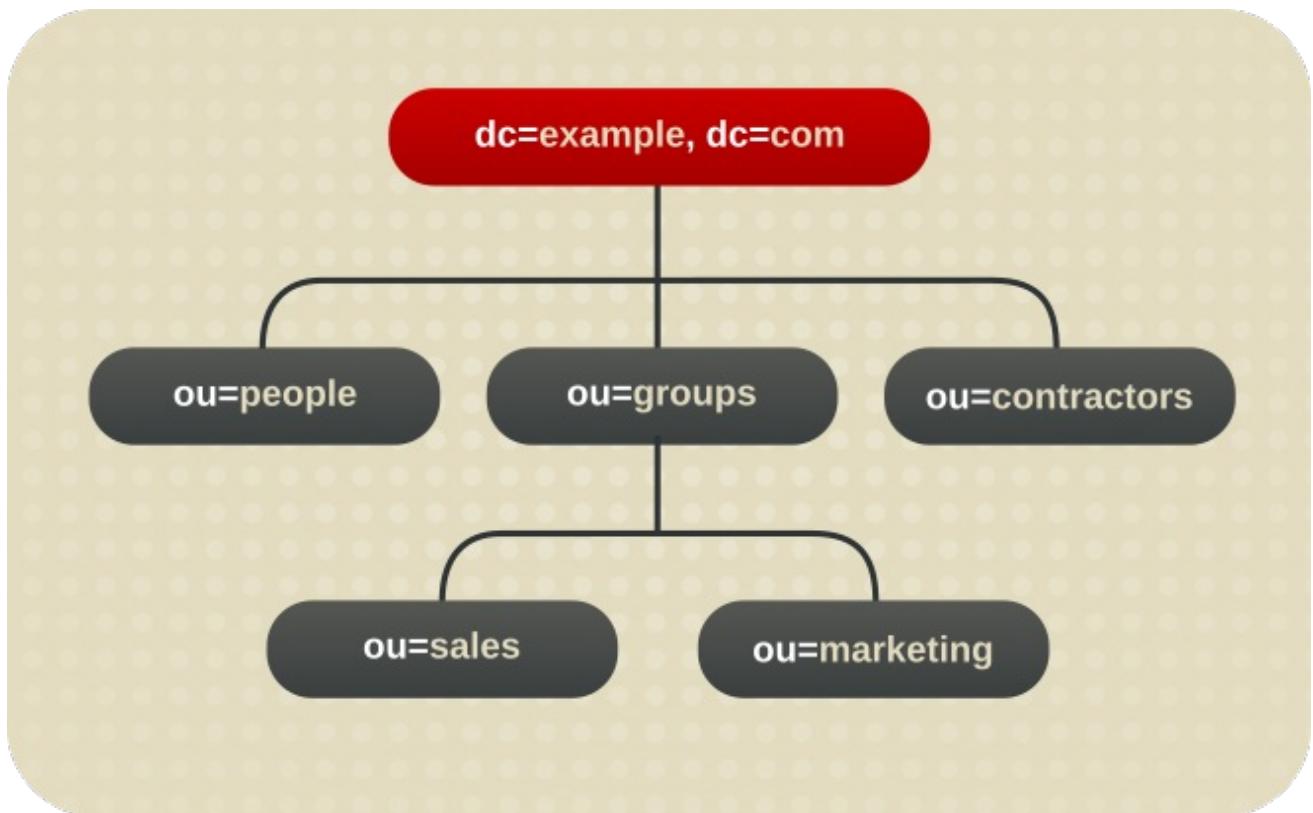


Figure 2.1. A Directory Tree with One Root Suffix

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

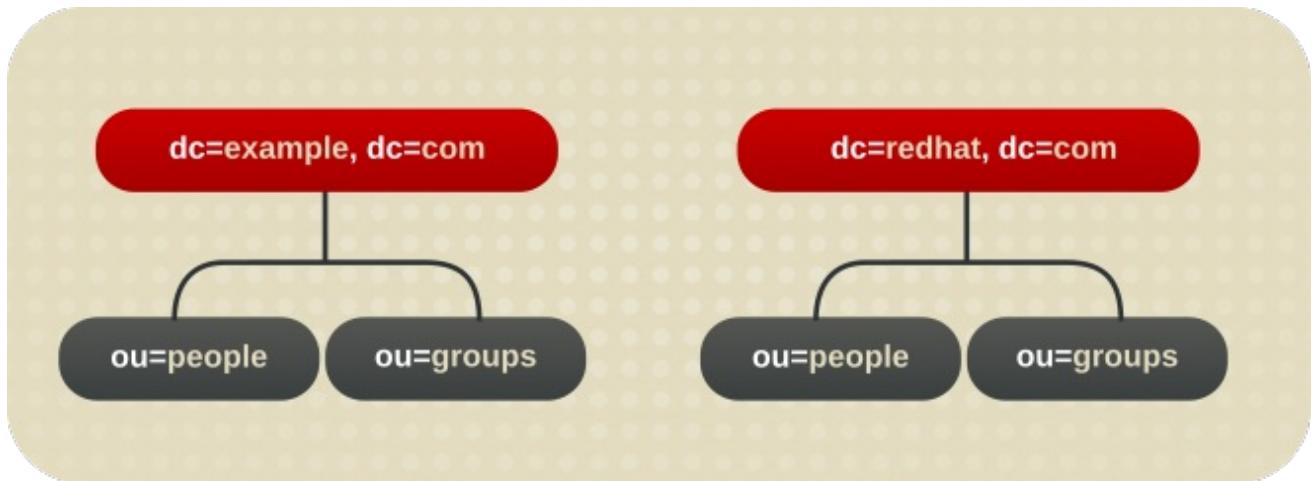
#### 2.1.1. Creating Suffixes

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

##### 2.1.1.1. Creating a Root Suffix

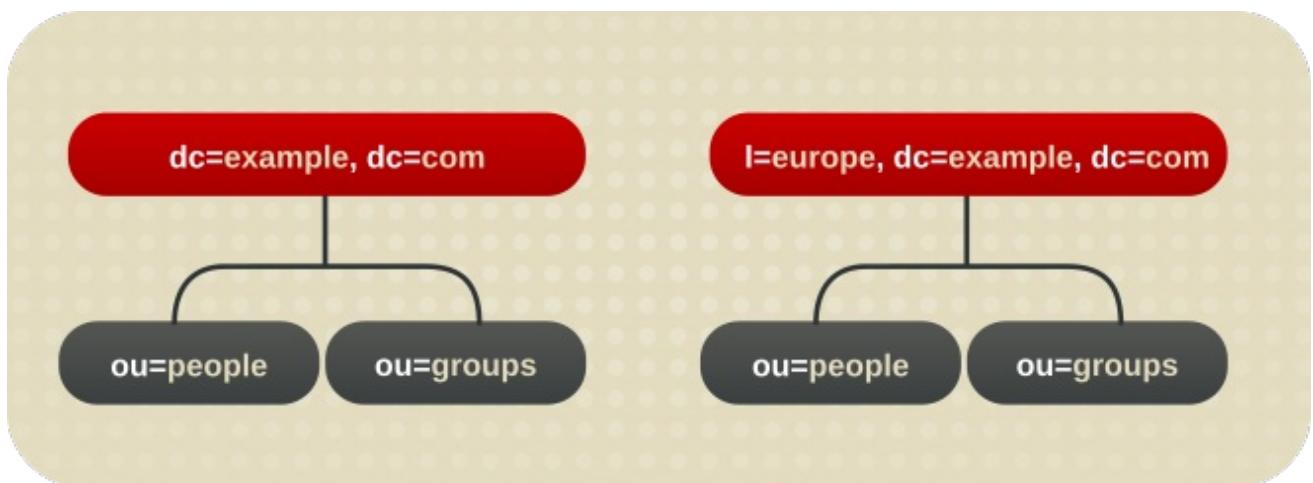
A directory can contain more than one root suffix. For example, an internet service provider that hosts several websites, one for **example.com** and one for **redhat.com**. In this scenario, two root suffixes are required. One corresponding to the **dc=example,dc=com** naming context and one corresponding to

the **dc=redhat,dc=com** naming context, as displayed in the following diagram:



**Figure 2.2. A Directory with Two Root Suffixes**

It is also possible to create root suffixes to exclude portions of the directory tree from search operations. For example, if the Example Corporation wants to exclude their European office from a search on the general Example Corporation directory. To implement this, the directory requires two root suffixes. One root suffix corresponds to the general Example Corporation directory tree, **dc=example,dc=com**, and one root suffix corresponds to the European branch of their directory tree, **l=europe,dc=example,dc=com**. From a client application's perspective, the directory tree looks as illustrated the following diagram:



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

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

#### 2.1.1.1. Creating a Root Suffix Using the Command Line

Use the **dsconf backend create** command to create a new root suffix:

1. Optional: Identify the suffixes and back end databases that are already in use:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

The name in parentheses is the back end database that stores the data of the corresponding suffix. You cannot use existing database names when you create the root suffix in the next step.

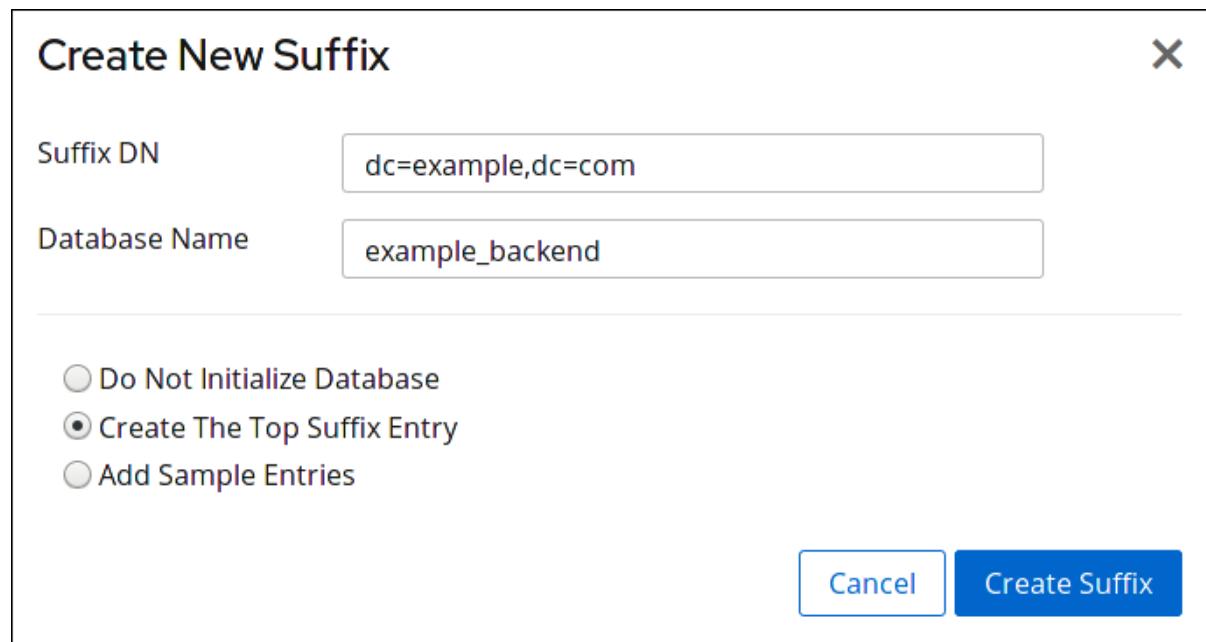
2. Create the **dc=example,dc=net** root suffix in the **example** back end database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend create \
--suffix="dc=example,dc=net" --be-name="example"
```

#### 2.1.1.2. Creating a Root Suffix Using the Web Console

To create a new root suffix using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Click **Create Suffix**.
5. Enter the suffix DN and back end name. For example:



6. Select **Create The Top Suffix Entry**.
7. Click **Create Suffix**.

#### 2.1.1.2. Creating a Sub-suffix

In certain scenarios, administrators want to store a branch of the directory tree in a separate database. For example, if the administrator creates the **l=europe,dc=example,dc=com** entry as a sub-suffix, this suffix is stored in a separate database. At the same time, the **dc=example,com** root suffix and all its sub-entries - except **l=europe,dc=example,dc=com** and subentries - are stored also in a separate database.

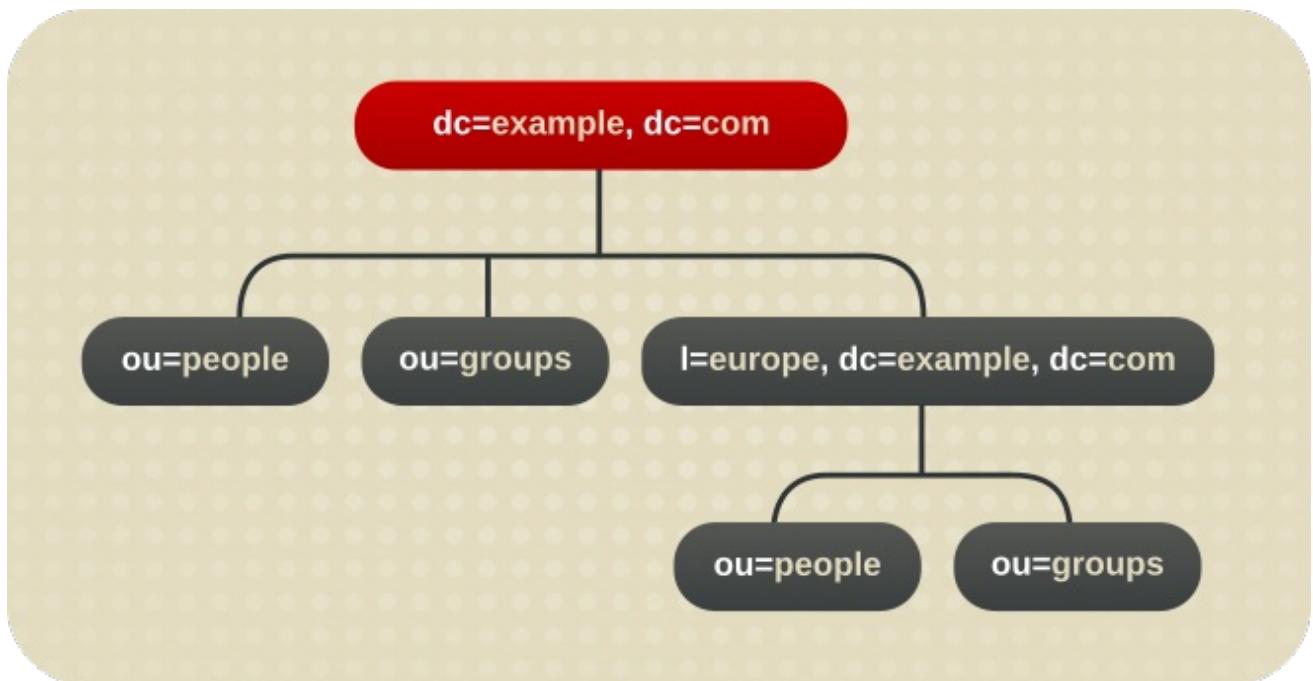


Figure 2.4. A Directory Tree with a Sub Suffix

#### 2.1.1.2.1. Creating a Sub-suffix Using the Command Line

Use the **dsconf backend create** command to create a new sub-suffix. For example, to create the **ou=People,dc=example,dc=com** sub-suffix in a new database called **people** under the **dc=example,dc=com** root suffix:

1. Optional: Identify the suffixes and back end databases that are already in use:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

The name in parentheses is the back end database that stores the data of the corresponding suffix. You cannot use existing database names when you create the sub-suffix in the next step.

2. Create the sub-suffix. For example, to create the **ou=People,dc=example,dc=com** sub-suffix along with the **example** back end database, enter:

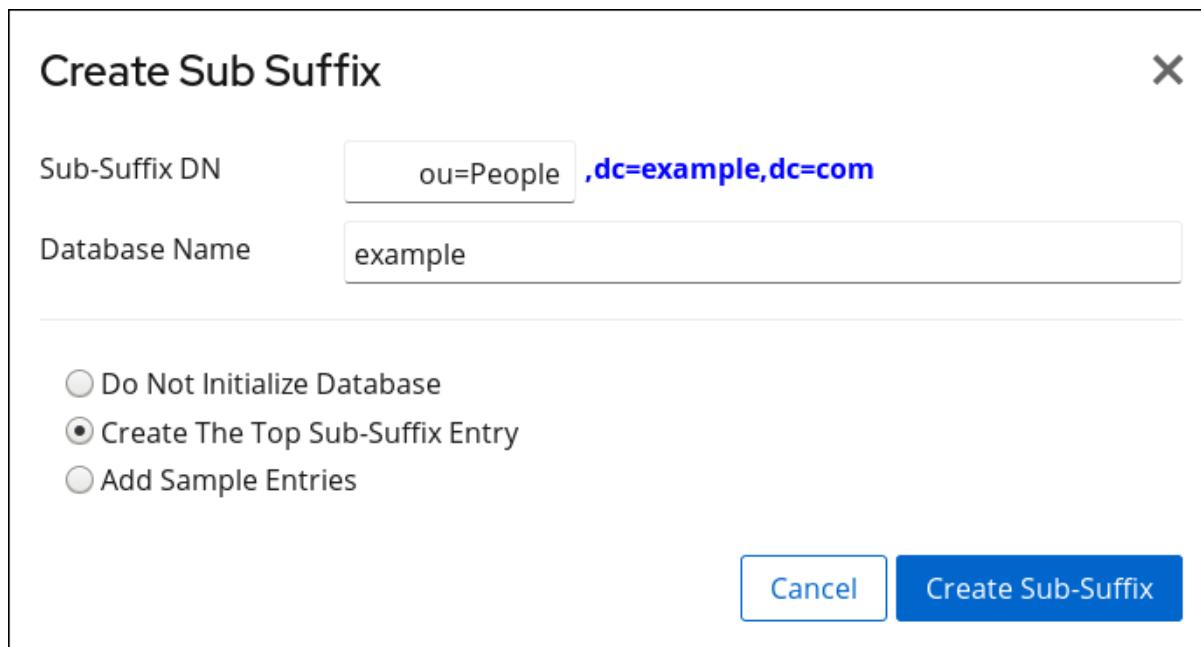
```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend create \
--suffix="ou=People,dc=example,dc=com" --be-name="example" \
--parent-suffix="dc=example,dc=com"
```

#### 2.1.1.2.2. Creating a Sub-suffix Using the Web Console

To create a new sub-suffix using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.

4. Select the suffix, under which you want to create the sub-suffix, click **Suffix Tasks**, and select **Create Sub-Suffix**.
5. Enter the sub-suffix DN and back end name. For example:



6. Select **Create The Top Sub-Suffix Entry**.

7. Click **Create Sub-Suffix**.

## 2.1.2. Maintaining Suffixes

### 2.1.2.1. Viewing the Default Naming Context

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

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

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

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



## IMPORTANT

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

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

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

### 2.1.2.2. Disabling a Suffix

In certain situations, a suffix in the directory needs to be disabled. If a suffix is disabled, the content of the database related to the suffix is no longer accessible by clients.

#### 2.1.2.2.1. Disabling a Suffix Using the Command Line

To disable a suffix using the command line, pass the back end database name to the **dsconf backend suffix set --disable** command. For example, to disable the ***o=test*** suffix:

1. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

This command displays the name of the back end database next to each suffix. You require the suffix's database name in the next step.

2. Disable the suffix:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend \
suffix set --disable "test_database"
```

### 2.1.2.3. Deleting a Suffix

If a suffix is no longer required, the administrator can delete it from the database.



## WARNING

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

### 2.1.2.3.1. Deleting a Suffix Using the Command Line

To delete a suffix using the command line, use the **dsconf backend delete** command. For example, to delete the **o=test** suffix:

1. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

This command displays the name of the back end database next to each suffix. You require the suffix's database name in the next step.

2. Delete the back end database and the corresponding suffix:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend delete
test_database
Deleting Backend cn=test_database,cn=ldbm database,cn=plugins,cn=config :
Type 'Yes I am sure' to continue: Yes I am sure
The database, and any sub-suffixes, were successfully deleted
```

### 2.1.2.3.2. Deleting a Suffix Using the Web Console

To delete a suffix using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the suffix, click **Suffix Tasks**, and select **Delete Suffix**.
5. Click **Yes** to confirm.

## 2.2. CREATING AND MAINTAINING DATABASES

After creating suffixes to organizing the directory data, create databases to contain data of that directory.



### NOTE

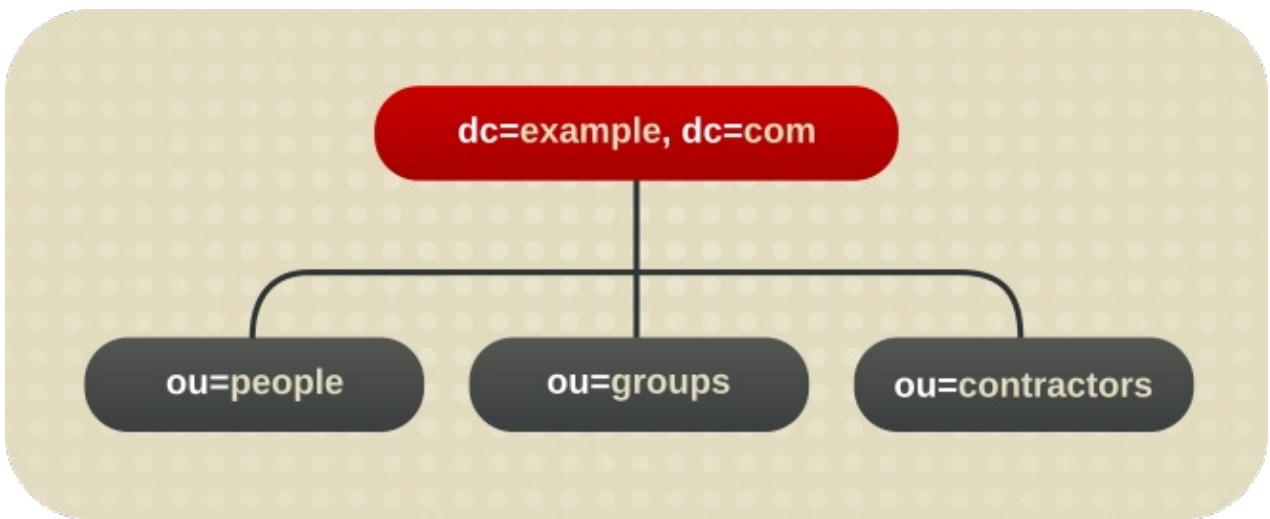
If you used the **dsconf** utility or the web console to create the suffix, Directory Server created the database automatically.

### 2.2.1. Creating Databases

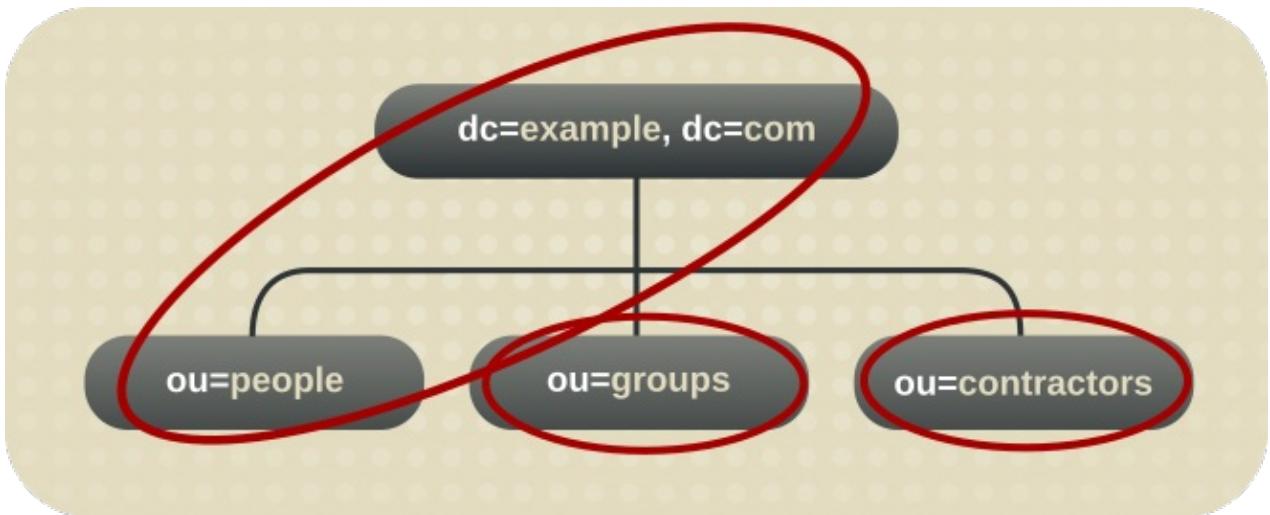
The directory tree can be distributed over multiple Directory Server databases. There are two ways to distribute data across multiple databases:

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

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



This division of the tree units corresponds to three databases, for example:

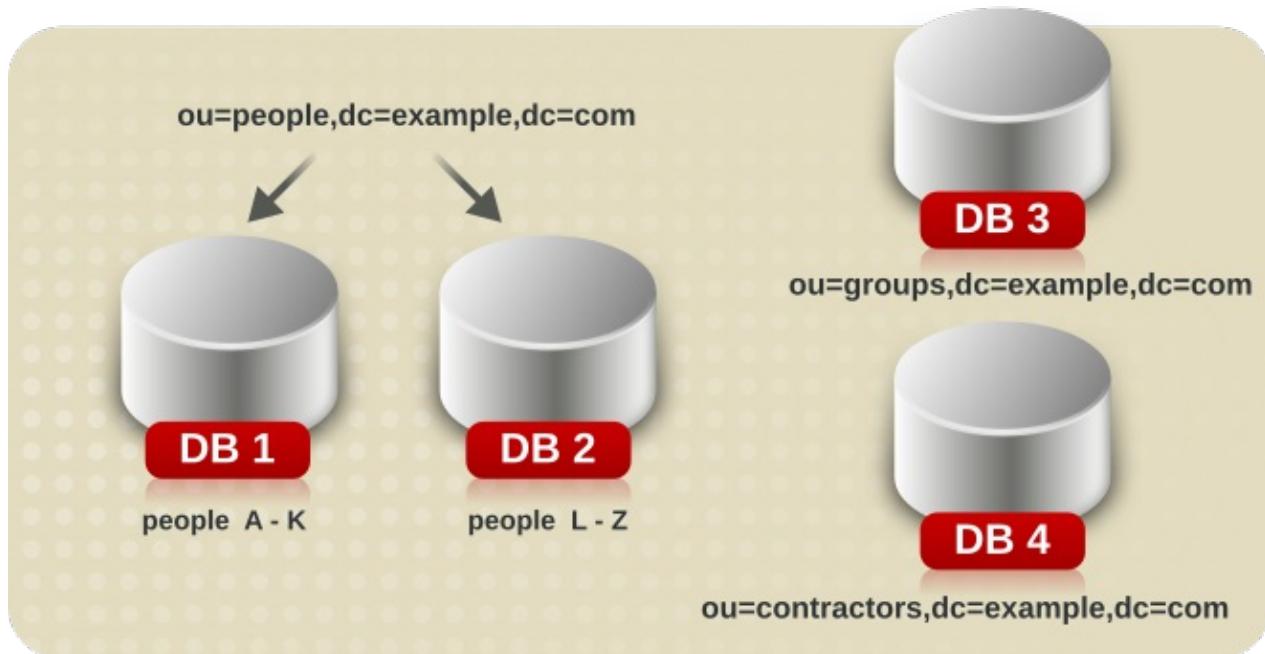


In this example, DB1 contains the data for **ou=people** and the data for **dc=example,dc=com**, so that clients can conduct searches based at **dc=example,dc=com**. However, DB2 only contains the data for **ou=groups**, and DB3 only contains the data for **ou=contractors**:



Multiple databases for one suffix.

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



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

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

### 2.2.1.1. Creating a New Database for a Single Suffix Using the Command Line

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

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

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=UserData,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: ou=people,dc=example,dc=com
```

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

2. Create a root or a sub-suffix, as described in [Section 2.1.1.1, “Creating a Root Suffix Using the Command Line”](#) and [Section 2.1.1.2, “Creating a Sub-suffix Using the Command Line”](#). The database name, given in the DN attribute, must correspond with the value in the **nsslapd-backend** attribute of the suffix entry.

### 2.2.1.2. Adding Multiple Databases for a Single Suffix

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



#### NOTE

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

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

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

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

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

To add a custom distribution function to a suffix:

1. Run **Idapmodify**.

```
# Idapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

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

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

```
- add: nsslapd-distribution-funct
  nsslapd-distribution-funct: distribution-function-name
```

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

## 2.2.2. Maintaining Directory Databases

### 2.2.2.1. Setting a Database in Read-Only Mode

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

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

#### 2.2.2.1.1. Setting a Database in Read-only Mode Using the Command Line

To set a database in read-only mode, use the **dsconf backend suffix set** command. For example, to set the database of the **o=test** suffix in read-only mode:

1. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

This command displays the name of the back end database next to each suffix. You require the suffix's database name in the next step.

2. Set the database in read-only mode:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --enable-readonly "test_database"
```

#### 2.2.2.1.2. Setting a Database in Read-only Mode Using the Web Console

To set a database in read-only mode:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the suffix entry.
5. Select **Database Read-Only Mode**.

6. Click **Save Configuration**.

### 2.2.2.2. Placing the Entire Directory Server in Read-Only Mode

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



#### WARNING

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



#### NOTE

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

#### 2.2.2.2.1. Placing the Entire Directory Server in Read-Only Mode Using the Command Line

To enable the read-only mode for Directory Server:

1. Set the **nsslapd\_READONLY** parameter to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-readonly=on
```

2. Restart the instance:

```
# dsctl instance_name restart
```

#### 2.2.2.2.2. Placing the Entire Directory Server in Read-Only Mode Using the Web Console

To enable the read-only mode for Directory Server:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the **Server Settings** entry.
4. On the **Advanced Settings** tab, select **Server Read-Only**.
5. Click **Save**.

### 2.2.2.3. Deleting a Database

If a suffix is no longer required, you can delete the database that stores the suffix.

#### 2.2.2.3.1. Deleting a Database Using the Command Line

To delete a database use the **dsconf backend delete** command. For example, to delete the database of the **o=test** suffix:

1. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

You require the name of the back end database, which is displayed next to the suffix, in the next step.

2. Delete the database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend delete
"test_database"
```

#### 2.2.2.3.2. Deleting a Database Using the Web Console

To delete a database using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the suffix to delete, click **Suffix Tasks**, and select **Delete Suffix**.
5. Click **Yes** to confirm.

#### 2.2.2.4. Changing the Transaction Log Directory

The transaction log enables Directory Server to recover the database, after an instance shut down unexpectedly. In certain situations, administrators want to change the path to the transaction logs. For example, to store them on a different physical disk than Directory Server database.



#### NOTE

To achieve higher performance, mount a faster disk to the directory that contains the transaction logs, instead of changing the location. For details, see the corresponding section in the [Red Hat Directory Server Performance Tuning Guide](#).

To change the location of the transaction log directory:

1. Stop Directory Server instance:

```
# dsctl instance_name stop
```

2. Create a new location for the transaction logs. For example:

```
# mkdir -p /srv/dirsrv/instance_name/db/
```

3. Set permissions to enable only Directory Server to access the directory:

```
# chown dirsrv:dirsrv /srv/dirsrv/instance_name/db/
# chmod 770 /srv/dirsrv/instance_name/db/
```

4. Remove all ***db.\**** files from the previous transaction log directory. For example:

```
# rm /var/lib/dirsrv/slapd-instance_name/db/*db.*
```

5. Move all ***log.\**** files from the previous to the new transaction log directory. For example:

```
# mv /var/lib/dirsrv/slapd-instance_name/db/log.* \
/srv/dirsrv/instance_name/db/
```

6. If SELinux is running in **enforcing** mode, set the ***dirsrv\_var\_lib\_t*** context on the directory:

```
# semanage fcontext -a -t dirsrv_var_lib_t /srv/dirsrv/instance_name/db/
# restorecon -Rv /srv/dirsrv/instance_name/db/
```

7. Edit the **/etc/dirsrv/slapd-*instance\_name*/dse.ldif** file, and update the ***nsslapd-db-logdirectory*** parameter under the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry. For example:

```
dn: cn=config,cn=ldbm database,cn=plugins,cn=config
...
nsslapd-db-logdirectory: /srv/dirsrv/instance_name/db/
```

8. Start the instance:

```
# dsctl instance_name start
```

## 2.3. CREATING AND MAINTAINING DATABASE LINKS

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

For more general information about chaining, see the chapter "Designing Directory Topology," in the *Red Hat Directory Server Deployment Guide*. [Section 21.8, "Monitoring Database Link Activity"](#) covers how to monitor database link activity.

### 2.3.1. Creating a New Database Link

The basic database link configuration requires the following information:

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



#### NOTE

If secure binds are required for simple password authentication ([Section 20.12.1, “Requiring Secure Binds”](#)), then any chaining operations will fail unless they occur over a secure connection. Using a secure connection (TLS and STARTTLS connections or SASL authentication) is recommended, anyway.

### 2.3.1.1. Creating a New Database Link Using the Command Line

To create a new database link, use the **dsconf chaining link-create** command. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-create --suffix="ou=Customers,dc=example,dc=com" --server-url="ldap://remote_server.example.com:389" --bind-mech="" --bind-dn="cn=proxy_user,cn=config" --bind-pw="password" "example_chain_name"
```

This creates a database link named **example\_chain\_name** for the **ou=Customers,dc=example,dc=com**. The link refers to the server **ldap://remote\_server.example.com:389** and uses the specified bind DN and password to authenticate. Because the **--bind-mech** is set empty, the link uses simple authentication.

To display additional settings you can set when you create the database link, see:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-create --help
```

For further details, see [Section 2.3.1.4, “Additional Information on Required Settings When Creating a Database Link”](#).

### 2.3.1.2. Creating a New Database Link Using the Web Console

To create a new database link:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Create a new suffix as described in [Section 2.1.1, “Creating Suffixes”](#).

5. Select the suffix, click **Suffix Tasks**, and select **Create Database Link**.
6. Fill the fields with the details about the connection to the remote server. For example:

**Create Database Link**

Link Sub-Suffix	ou=Example ,dc=example,dc=com	X
Link Database Name		
Example Link		
Remote Server URL(s)		
ldap://server.example.com		
<input checked="" type="checkbox"/> Use StartTLS		
Remote Server Bind DN		
cn=user,ou=People,dc=example,dc=com		
Bind DN Credentials		
*****		
Confirm Password		
*****		
Bind Method	Simple	▼
<a href="#">Cancel</a>		<a href="#">Create Database Link</a>

For further details, see [Section 2.3.1.4, “Additional Information on Required Settings When Creating a Database Link”](#).

7. Click **Create Database Link**.

### 2.3.1.3. Managing the Default Configuration for New Database Links

To create a set of configuration settings, Directory Server uses when you create a new database link:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def ...
```

For a list of parameters you can set and their descriptions, see:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def --help
```

To display the current default values:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-get-def
```

### 2.3.1.4. Additional Information on Required Settings When Creating a Database Link

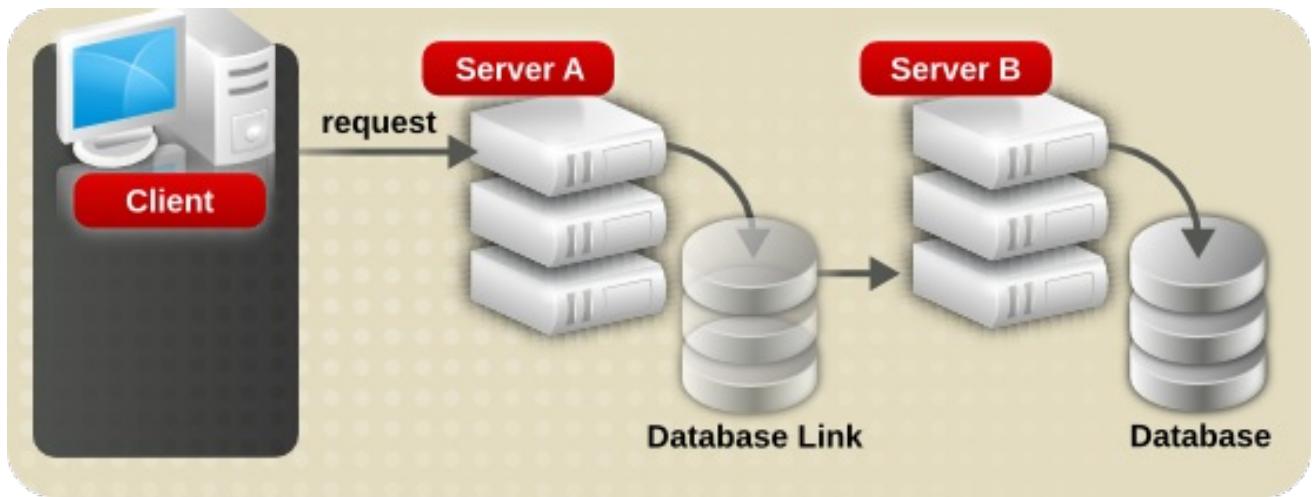
#### Suffix Information

The suffix defines the suffix that is managed by the database link.

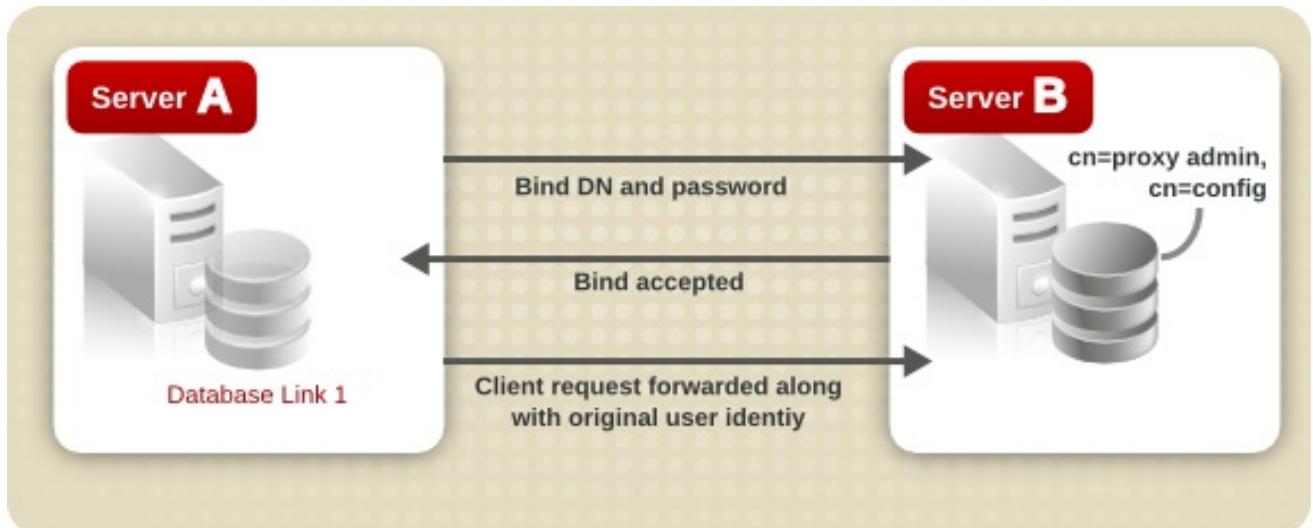
#### Bind Credentials

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

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 and password:



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



## WARNING

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

For more information on ACIs, see [Chapter 18, Managing Access Control](#).



## NOTE

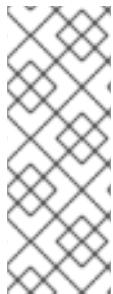
When a database link is used by a client application to create or modify entries, the attributes ***creatorsName*** and ***modifiersName*** do not reflect the real creator or modifier of the entries. These attributes contain the name of the administrative user granted proxied authorization rights on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

Providing bind credentials involves the following steps on the remote server:

1. Create an administrative user, such as ***cn=proxy\_user,cn=config***, for the database link. For information on adding entries, see [Chapter 3, Managing Directory Entries](#).
2. Provide proxy access rights for the administrative user created in the previous step on the subtree chained to by the database link. For more information on configuring ACIs, see [Chapter 18, Managing Access Control](#)

For example, the following ACI grants read-only access to the ***cn=proxy\_admin,cn=config*** user to access data contained on the remote server only within the subtree where the ACI is set.

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



## NOTE

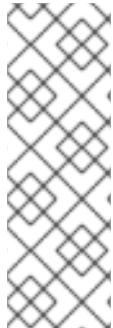
When a user binds to a database link, the user's identity is sent to the remote server. Access controls are always evaluated on the remote server. For the user to modify or write data successfully to the remote server, set up the correct access controls on the remote server. For more information about how access controls are evaluated in the context of chained operations, see [Section 2.3.3, "Database Links and Access Control Evaluation"](#).

## LDAP URL

On the server containing the database link, identify the remote server that the database link connects with using an *LDAP URL*. Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It takes the form ***ldap://host\_name:port***.

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

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



### NOTE

TLS has to be enabled on the local Directory Server and the remote Directory Server to be chained over TLS. For more information on enabling TLS, see [Section 9.4, "Enabling TLS"](#).

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

## Bind Mechanisms

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

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

- Over the standard LDAP port
- Over a dedicated LDAPS port
- Using STARTTLS, which is a secure connection over a standard port



### NOTE

If secure binds are required for simple password authentication ([Section 20.12.1, "Requiring Secure Binds"](#)), then any chaining operations will fail unless they occur over a secure connection (TLS and STARTTLS connections or SASL authentication) is recommended, anyway.

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

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

- *empty*: If there is no bind mechanism set, then the server performs simple authentication and requires a bind DN and password.
- *EXTERNAL*: This uses an TLS certificate to authenticate the farm server to the remote server. Either the farm server URL must be set to the secure URL (**ldaps**) or the **nsUseStartTLS** attribute must be set to **on**.

Additionally, the remote server must be configured to map the farm server's certificate to its bind identity, as described in the *certmap.conf* section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

- **DIGEST-MD5:** This uses SASL authentication with DIGEST-MD5 encryption. As with simple authentication, this requires the ***nsMultiplexorBindDN*** and ***nsMultiplexorCredentials*** attributes to give the bind information.
- **GSSAPI:** This uses Kerberos-based authentication over SASL.

The farm server must be configured with a Kerberos keytab, and the remote server must have a defined SASL mapping for the farm server's bind identity. Setting up Kerberos keytabs and SASL mappings is described in [Section 9.10, "Setting up SASL Identity Mapping"](#).



#### NOTE

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



#### NOTE

If SASL is used, then the local server must also be configured to chain the SASL and password policy components. Add the components for the database link configuration, as described in [Section 2.3.2, "Configuring the Chaining Policy"](#).

### 2.3.2. Configuring the Chaining Policy

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

#### 2.3.2.1. Chaining Component Operations

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

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

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

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

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

##### ACI plug-in

This plug-in implements access control. Operations used to retrieve and update ACI attributes are not chained because it is not safe to mix local and remote ACI attributes. However, requests used to retrieve user entries may be chained by setting the chaining components attribute:

nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config

Permissions: Read, search, and compare

### Resource limit component

This component sets server limits depending on the user bind DN. Resource limits can be applied on remote users if the resource limitation component is allowed to chain. To chain resource limit component operations, add the chaining component attribute:

nsActiveChainingComponents: cn=resource limits,cn=components,cn=config

Permissions: Read, search, and compare

### Certificate-based authentication checking component

This component is used when the external bind method is used. It retrieves the user certificate from the database on the remote server. Allowing this component to chain means certificate-based authentication can work with a database link. To chain this component's operations, add the chaining component attribute:

nsActiveChainingComponents: cn=certificate-based authentication,cn=components,cn=config

Permissions: Read, search, and compare

### Password policy component

This component is used to allow SASL binds to the remote server. Some forms of SASL authentication require authenticating with a user name and password. Enabling the password policy allows the server to verify and implement the specific authentication method requested and to apply the appropriate password policies. To chain this component's operations, add the chaining component attribute:

nsActiveChainingComponents: cn=password policy,cn=components,cn=config

Permissions: Read, search, and compare

### SASL component

This component is used to allow SASL binds to the remote server. To chain this component's operations, add the chaining component attribute:

nsActiveChainingComponents: cn=password policy,cn=components,cn=config

Permissions: Read, search, and compare

### Referential Integrity plug-in

This plug-in ensures that updates made to attributes containing DNs are propagated to all entries that contain pointers to the attribute. For example, when an entry that is a member of a group is deleted, the entry is automatically removed from the group. Using this plug-in with chaining helps simplify the management of static groups when the group members are remote to the static group definition. To chain this component's operations, add the chaining component attribute:

nsActiveChainingComponents: cn=referential integrity postoperation,cn=plugins,cn=config

Permissions: Read, search, and compare

## Attribute Uniqueness plug-in

This plug-in checks that all the values for a specified attribute are unique (no duplicates). If this plug-in is chained, it confirms that attribute values are unique even on attributes changed through a database link. To chain this component's operations, add the chaining component attribute:

```
nsActiveChainingComponents: cn=attribute uniqueness,cn=plugins,cn=config
```

Permissions: Read, search, and compare

## Roles component

This component chains the roles and roles assignments for the entries in a database. Chaining this component maintains the roles even on chained databases. To chain this component's operations, add the chaining component attribute:

```
nsActiveChainingComponents: cn=roles,cn=components,cn=config
```

Permissions: Read, search, and compare



### NOTE

The following components cannot be chained:

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

When enabling the Referential Integrity plug-in on servers issuing chaining requests, be sure to analyze performance, resource, and time needs as well as integrity needs. Integrity checks can be time-consuming and draining on memory and CPU. For further information on the limitations surrounding ACIs and chaining, see [Section 18.5, "Limitations of ACIs"](#).

### 2.3.2.1.1. Chaining Component Operations Using the Command Line

To add a component allowed to chain:

1. Specify the components to include in chaining. For example, to configure that the referential integrity component can chain operations:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set \
--add-comp="cn=referential integrity postoperation,cn=components,cn=config"
```

See [Section 2.3.2.1, "Chaining Component Operations"](#) for a list of the components which can be chained.

2. Restart the instance:

```
# dsctl instance_name restart
```

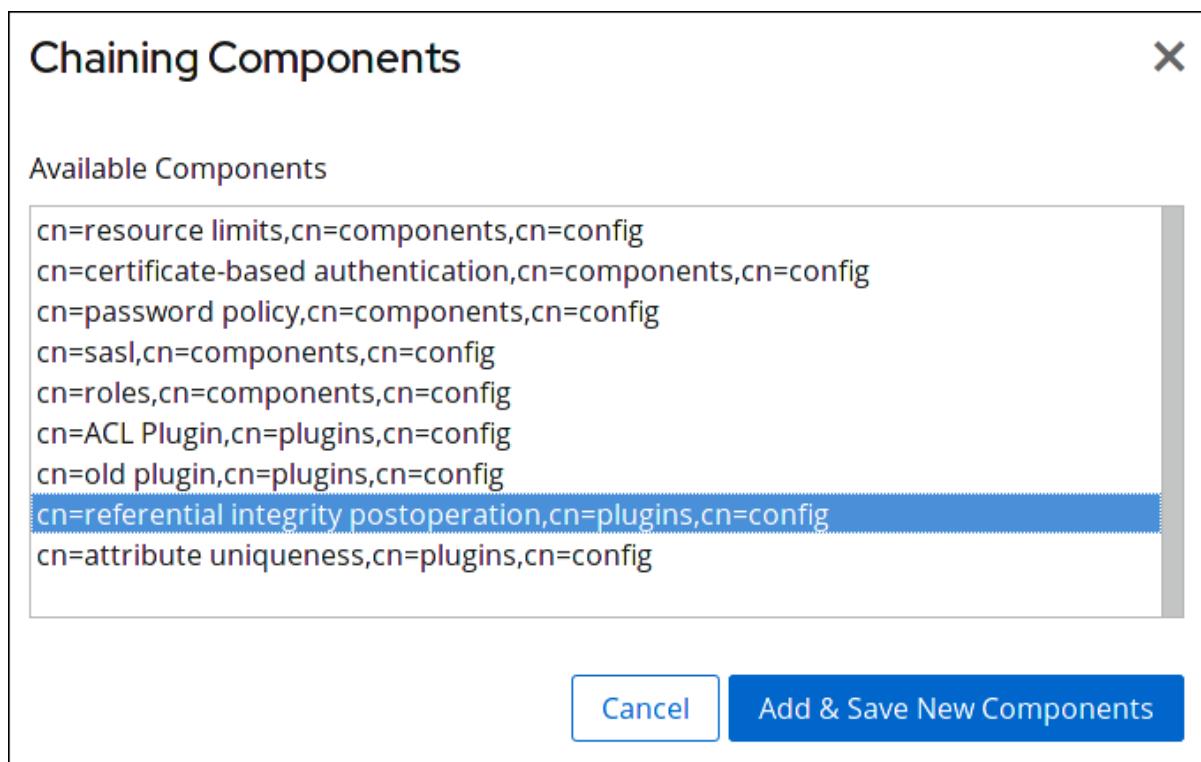
3. Create an ACI in the suffix on the remote server to which the operation will be chained. For example, to create an ACI for the Referential Integrity plug-in:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr "")(target="ldap:///ou=customers,l=us,dc=example,dc=com")
(version 3.0; acl "Reflint Access for chaining"; allow
(read,write,search,compare) userdn = "ldap:///cn=referential
integrity postoperation,cn=plugins,cn=config";)
```

### 2.3.2.1.2. Chaining Component Operations Using the Web Console

To add a component allowed to chain:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the **Chaining Configuration** entry.
5. Click the **Add** button below the **Components to Chain** field.
6. Select the component and click **Add & Save Components**.



7. Click **Save**.
8. Create an ACI in the suffix on the remote server to which the operation will be chained. For example, to create an ACI for the Referential Integrity plug-in:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr "")(target="ldap:///ou=customers,l=us,dc=example,dc=com")
(version 3.0; acl "Reflint Access for chaining"; allow
(read,write,search,compare) userdn = "ldap:///cn=referential
integrity postoperation,cn=plugins,cn=config");
```

### 2.3.2.2. Chaining LDAP Controls

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

- *Virtual List View (VLV)*. This control provides lists of parts of entries rather than returning all entry information.
- *Server-side sorting*. This control sorts entries according to their attribute values, usually using a specific matching rule.
- *Dereferencing*. This control tracks back over references in entry attributes in a search and pulls specified attribute information from the referenced entry and returns it with the rest of the search results.
- *Managed DSA*. This control returns smart referrals as entries, rather than following the referral, so the smart referral itself can be changed or deleted.
- *Loop detection*. This control keeps track of the number of times the server chains with another server. When the count reaches the configured number, a loop is detected, and the client application is notified. For more information about using this control, see [Section 2.4.3, “Detecting Loops”](#).



#### NOTE

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

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

**Table 2.1. 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

Control Name	OID
Dereferencing searches	1.3.6.1.4.1.4203.666.5.16

### 2.3.2.2.1. Chaining LDAP Controls Using the Command Line

To chain LDAP controls, use the **dsconf chaining config-set --add-control** command. For example, to forward the virtual list view control:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining \
config-set --add-control="2.16.840.1.113730.3.4.9"
```

If clients of Directory Server create their own controls and their operations should be chained to remote servers, add the object identifier (OID) of the custom control.

For a list of LDAP controls that can be chained and their OIDs, see [Table 2.1, “LDAP Controls and Their OIDs”](#).

### 2.3.2.2.2. Chaining LDAP Controls Using the Web Console

To chain LDAP controls using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the **Chaining Configuration** entry.
5. Click the **Add** button below the **Forwarded LDAP Controls** field.
6. Select the LDAP control and click **Add & Save New Controls**.

**Chaining LDAP Controls**

**Available LDAP Controls**

- 2.16.840.1.113730.3.4.14
- 2.16.840.1.113730.3.4.15
- 2.16.840.1.113730.3.4.16
- 2.16.840.1.113730.3.4.17
- 2.16.840.1.113730.3.4.18
- 2.16.840.1.113730.3.4.19
- 2.16.840.1.113730.3.4.20
- 2.16.840.1.113730.3.4.3
- 2.16.840.1.113730.3.4.4**
- 2.16.840.1.113730.3.4.5

**Cancel** **Add & Save New Controls**

If clients of Directory Server create their own controls and their operations should be chained to remote servers, add the object identifier (OID) of the custom control.

For a list of LDAP controls that can be chained and their OIDs, see [Table 2.1, “LDAP Controls and Their OIDs”](#).

7. Click **Save**.

### 2.3.3. Database Links and Access Control Evaluation

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

- Not all types of access control can be used.

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

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



#### NOTE

Directory Server supports both IPv4 and IPv6 IP addresses.

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

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

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

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

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

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

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



#### NOTE

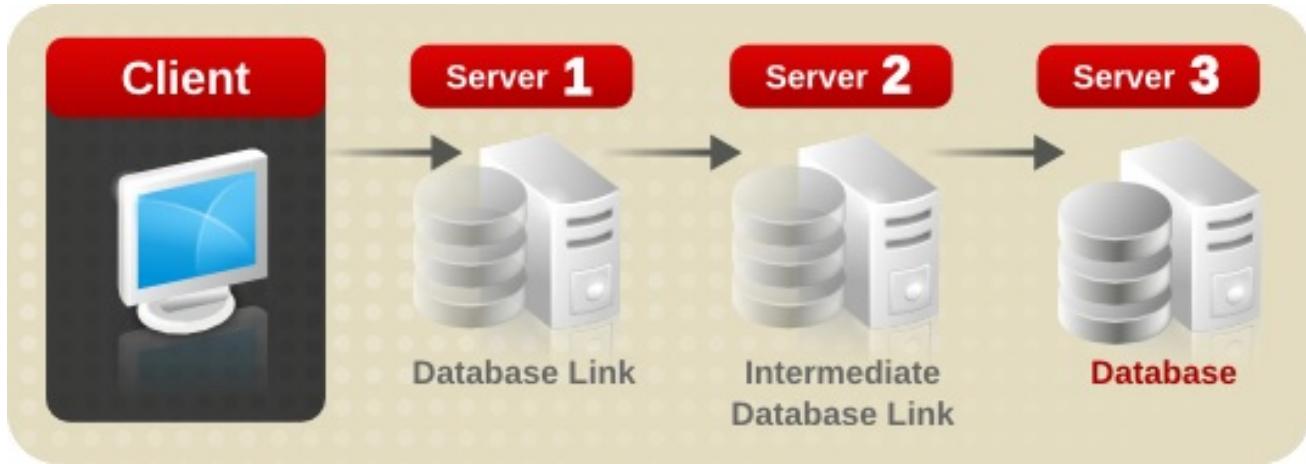
By default, access controls set on the server containing the database link are not evaluated. To override this default, use the **nsCheckLocalACI** attribute in the **cn=database\_link, cn=chaining database,cn=plugins,cn=config** entry. However, evaluating access controls on the server containing the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

## 2.4. CONFIGURING CASCADING CHAINING

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

### 2.4.1. Overview of Cascading Chaining

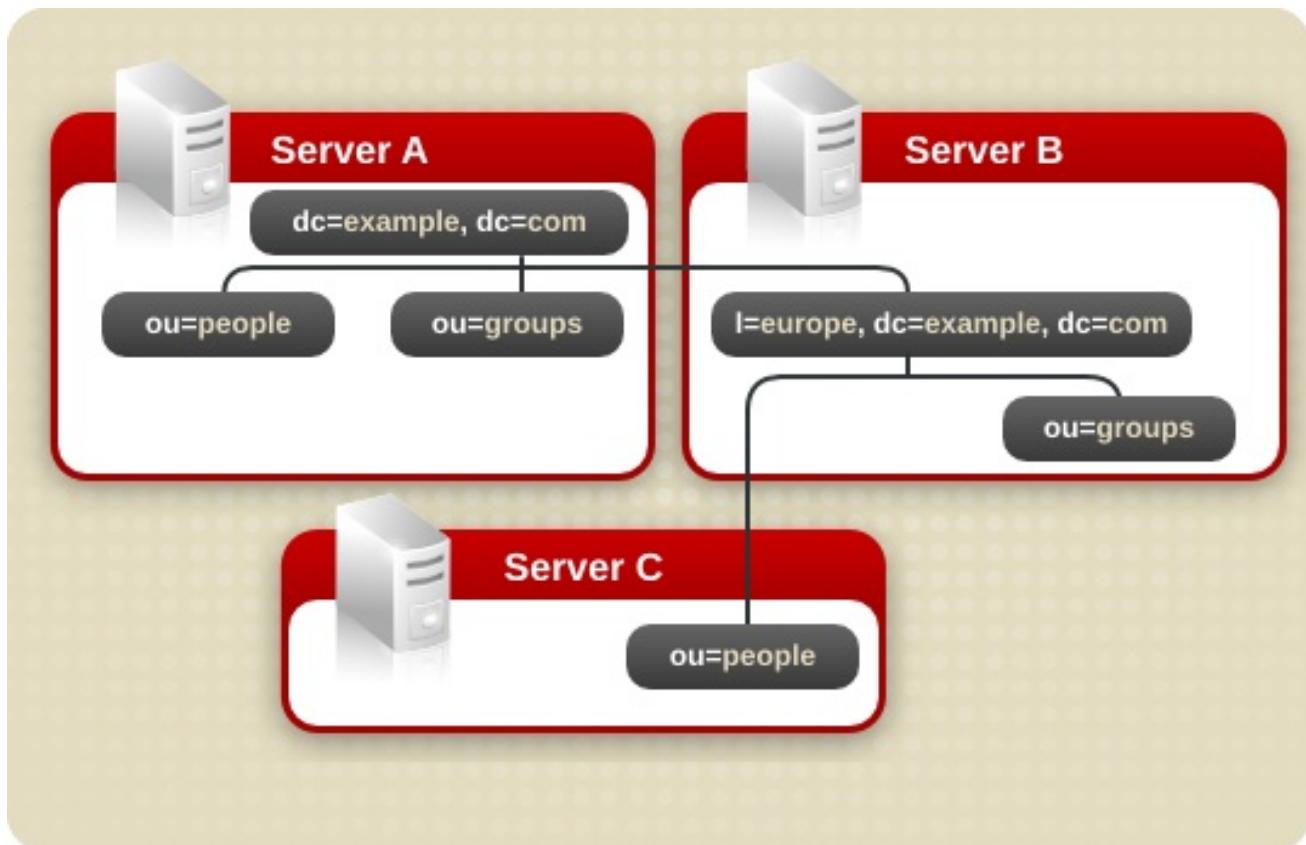
Cascading chaining occurs when more than one hop is required for the directory to process a client application's request.



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

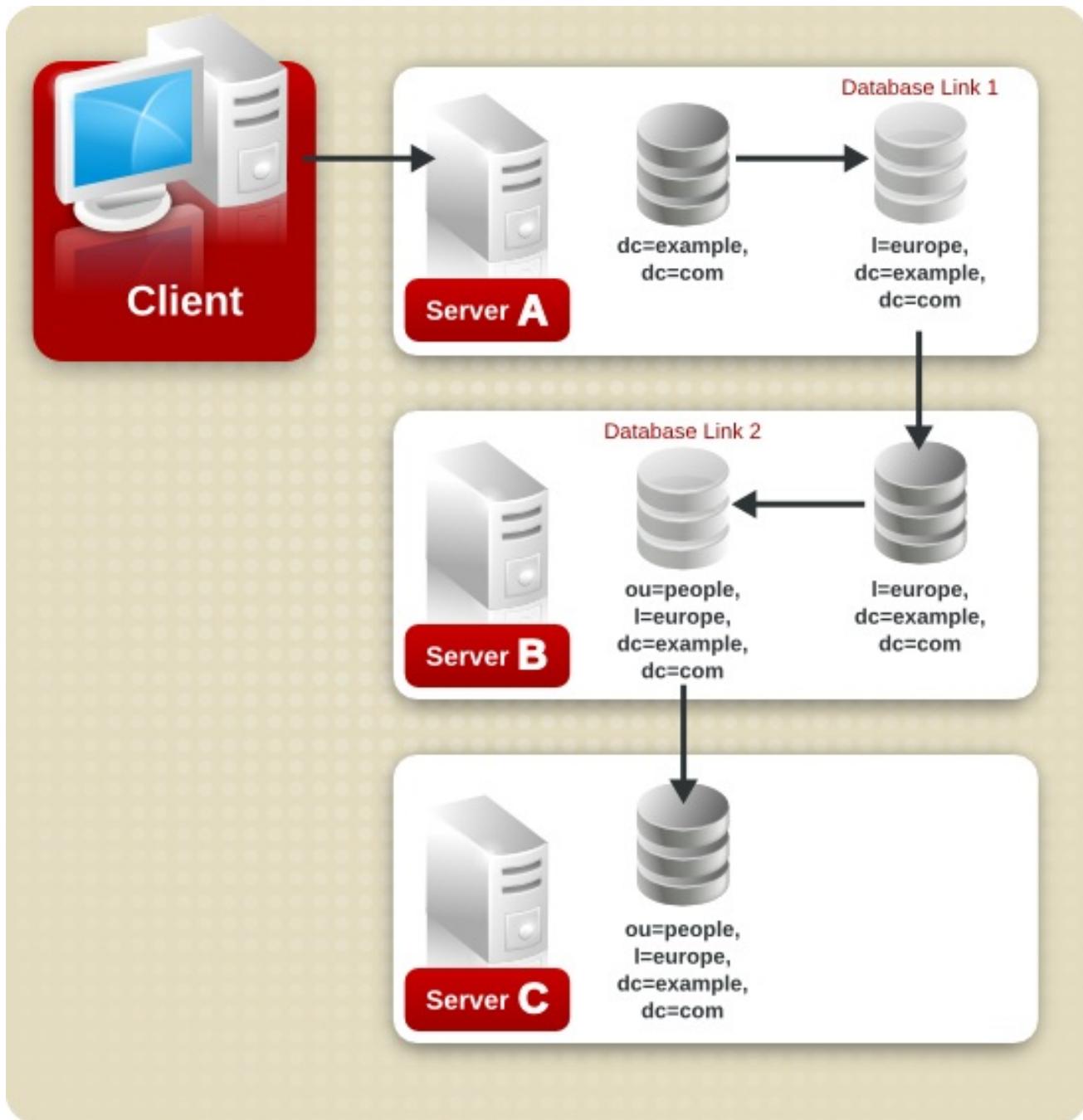
During a normal operation request, a client binds to the server, and then any ACIs applying to that client are evaluated. With cascading chaining, the client bind request is evaluated on Server 1, but the ACIs applying to the client are evaluated only after the request has been chained to the destination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

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



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

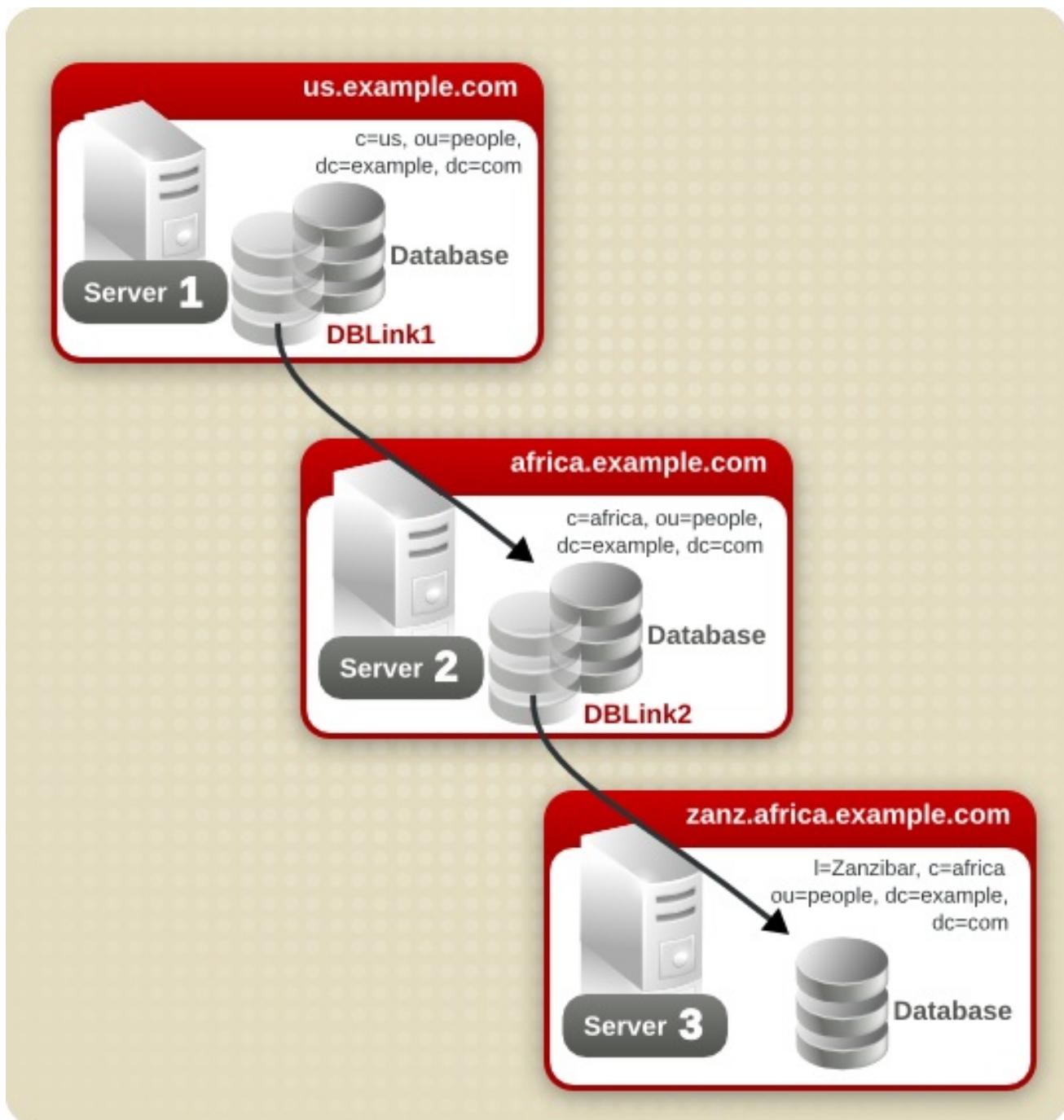
With cascading configured on servers A, B, and C, a client request targeted at the **ou=people,l=europe,dc=example,dc=com** entry would be routed by the directory as follows:



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=europa,dc=example,dc=com`** branch. Because at least two hops are required for the directory to service the client request, this is considered a cascading chain.

#### 2.4.2. Configuring Cascading Chaining Using the Command Line

This section provides an example of how to configure cascading chaining with three servers as shown in the following diagram:



### Configuration Steps on Server 1

1. Create the suffix **`c=africa,ou=people,dc=example,dc=com`**:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com create --parent-suffix="ou=people,dc=example,dc=com" --suffix="c=africa,ou=people,dc=example,dc=com"
```

2. Create the **DBLink1** database link:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com chaining link-create --suffix="c=africa,ou=people,dc=example,dc=com" --server-url="ldap://africa.example.com:389/" --bind-mech="" --bind-dn="cn=server1 proxy admin,cn=config" --bind-pw="password" --check-aci="off" "DBLink1"
```

3. Enable loop detection:

■

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com chaining config-set --add-control="1.3.6.1.4.1.1466.29539.12"
```

## Configuration Steps on Server 2

1. Create a proxy administrative user on server 2 for server 1 to use for proxy authorization:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server2.example.com -x
dn: cn=server1 proxy admin,cn=config
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: server1 proxy admin
sn: server1 proxy admin
userPassword: password
description: Entry for use by database links
```



### IMPORTANT

For security reasons, do not use the **cn=Directory Manager** account.

2. Create the suffix **I=Zanzibar,c=africa,ou=people,dc=example,dc=com**:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com create --parent-suffix="c=africaou=people,dc=example,dc=com" --
suffix="I=Zanzibar,c=africa,ou=people,dc=example,dc=com"
```

3. Create the **DBLink2** database link:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com chaining link-create --
suffix="I=Zanzibar,c=africa,ou=people,dc=example,dc=com" --server-url="ldap://zanz.africa.example.com:389/" --bind-mech="" --bind-dn="server2 proxy admin,cn=config" --bind-pw="password" --check-aci="on" "DBLink2"
```

Because the **DBLink2** link is the intermediate database link in the cascading chaining configuration, enable the ACL check to allow the server to check whether it should allow the client and proxy administrative user access to the database link.

4. Enable loop detection:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com chaining config-set --add-control="1.3.6.1.4.1.1466.29539.12"
```

5. Enable the proxy authorization control:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com chaining config-set --add-control="2.16.840.1.113730.3.4.12"
```

6. Add the local proxy authorization ACI:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server2.example.com -x
dn: c=africa,ou=people,dc=example,dc=com
```

```

changetype: modify
add: aci
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";)

```

- Add an ACI that enables users in **c=us,ou=people,dc=example,dc=com** on server 1 who have a **uid** attribute set, to perform any type of operation on the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** suffix tree on server 3:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server2.example.com -x
dn: c=africa,ou=people,dc=example,dc=com
changetype: modify
add: aci
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";)

```

If there are users on server 3 under a different suffix that will require additional rights on server 3, it is necessary to add additional client ACIs on server 2.

### Configuration Steps on Server 3

- Create a proxy administrative user on server 3 for server 2 to use for proxy authorization:

```

# ldapadd -D "cn=Directory Manager" -W -p 389 -h server3.example.com -x
dn: cn=server2 proxy admin,cn=config
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: server2 proxy admin
sn: server2 proxy admin
userPassword: password
description: Entry for use by database links

```



#### IMPORTANT

For security reasons, do not use the **cn=Directory Manager** account.

- Add the local proxy authorization ACI:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server3.example.com -x
dn: l=Zanzibar,ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "")(version 3.0; acl "Proxied authorization
  for database links"; allow (proxy) userdn = "ldap:///cn=server2
  proxy admin,cn=config";)

```

- Add an ACI that enables users in **c=us,ou=people,dc=example,dc=com** on server 1 who have a **uid** attribute set, to perform any type of operation on the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** suffix tree on server 3:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server3.example.com -x
dn: l=Zanzibar,ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr ="(objectClass=*)") (target="l=Zanzibar,c=africa,ou=people,dc=example,dc=com")
    (version 3.0; acl "Client authentication for database link users"; allow (all)
    userdn = "ldap:///uid=*,c=us,ou=people,dc=example,dc=com";)
```

If there are users on server 3 under a different suffix that will require additional rights on server 3, it is necessary to add additional client ACIs on server 2.

The cascading chaining configuration is now set up. This cascading configuration enables a user to bind to server 1 and modify information in the **l=Zanzibar,c=africa,ou=people,dc=example,dc=com** branch on server 3. Depending on your security needs, it can be necessary to provide more detailed access control.

### 2.4.3. Detecting Loops

An LDAP control included with Directory Server prevents loops. When first attempting to chain, the server sets this control to 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.

To use the control, add the **1.3.6.1.4.1.1466.29539.12** OID. For details about adding an LDAP control, see [Section 2.3.2.2, “Chaining LDAP Controls”](#). If the control is not present in the configuration file of each database link, loop detection will not be implemented.

The number of hops allowed is defined using the **nsHopLimit** parameter. By default, the parameter is set to **10**. For example, to set the hop limit of the **example** chain to **5**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-set --hop-limit 5
example
```

## 2.5. USING REFERRALS

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

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

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

### 2.5.1. Starting the Server in Referral Mode

Referrals are used to redirect client applications to another server while the current server is unavailable

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

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

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

- **/etc/dirsrv/slapd-*instance\_name*** is the directory where the Directory Server configuration files are. This is the default location on Red Hat Enterprise Linux.
- *port* is the optional port number of Directory Server to start in referral mode.
- *referral\_url* is the referral returned to clients. The format of an LDAP URL is covered in [Appendix C, LDAP URLs](#).

## 2.5.2. Setting Default Referrals

Directory Server returns default referrals to client applications that submit operations on a DN not contained within any of the suffixes maintained by the directory. The following procedures describe setting a default referral for the directory using the command line.

### 2.5.2.1. Setting a Default Referral Using the Command Line

Use the **dsconf config replace** command, to set the default referral in the ***nsslapd-referral*** parameter. For example, to set **ldap://directory.example.com/** as the default referral:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-referral="ldap://directory.example.com/"
```

## 2.5.3. Creating Smart Referrals

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

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

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

### 2.5.3.1. Creating Smart Referrals Using the Command Line

To create a smart referral, create the relevant directory entry with the **referral** object class and set the **ref** attribute to the referral LDAP URL.

For example, to create a smart referral named **uid=user,ou=people,dc=example,dc=com** that refers to **ldap://directory.europe.example.com/cn=user,ou=people,l=europe,dc=example,dc=com**:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server2.example.com -x  
dn: uid=user,ou=people,dc=example,dc=com
```

```

objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: referral
sn: user
uid: user
cn: user
ref: ldap://directory.europe.example.com/cn=user,ou=people,l=europe,dc=example,dc=com

```



#### NOTE

Directory Server ignores any information after a space in an LDAP URL. For this reason, use **%20** instead of spaces in LDAP URLs used as a referral.

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

### 2.5.4. Creating Suffix Referrals

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



#### WARNING

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

#### 2.5.4.1. Creating Suffix Referrals Using the Command Line

To create a suffix referral:

1. Optionally, create a root or sub-suffix, if it does not already exist. For details, see [Section 2.1.1, "Creating Suffixes"](#).
2. Add the referral to the suffix. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com suffix set --add-referral="ldap://directory.example.com/" database_name
```

#### 2.5.4.2. Creating Suffix Referrals Using the Web Console

To create a suffix referral:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.

3. Open the **Database** menu.
4. Optionally, create a root or sub-suffix, if it does not already exist. For details, see [Section 2.1.1, "Creating Suffixes"](#).
5. Select the suffix in the list, and open the **Referrals** tab.
6. Click **Create Referral**.
7. Fill the fields to create the referral URL.

**Add Database Referral**

Protocol	<input type="text" value="ldaps://"/>
Host Name	<input type="text" value="server.example.com"/>
Port Number	<input type="text" value="636"/>
Suffix	<input type="text" value="dc=ldap,dc=example,dc=com"/>
Attributes	<input type="text"/>
Filter	<input type="text"/>
Scope	<input type="text"/>
Computed Referral	<input type="text" value="ldaps://server.example.com:636/dc%3Dldap%2Cdc%3Dexam"/>
<input type="button" value="Cancel"/> <input type="button" value="Create Referral"/>	

8. Click **Create Referral**.

## 2.6. VERIFYING THE INTEGRITY OF BACK-END DATABASES

The **dsctl dbverify** command enables administrators to verify the integrity of back-end databases. For example, to verify the **userroot** database:

1. Optionally, list the back-end databases of the instance:

```
dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

You need the name of the database in a later step.

2. Stop the Directory Server instance:

```
# dsctl instance_name stop
```

3. Verify the database:

```
# dsctl instance_name dbverify userroot  
[04/Feb/2020:13:11:02.453624171 +0100] - INFO -  
ldbm_instance_config_cachememsize_set - force a minimal value 512000  
[04/Feb/2020:13:11:02.465339507 +0100] - WARN -  
ldbm_instance_add_instance_entry_callback - ldbm instance userroot already exists  
[04/Feb/2020:13:11:02.468060144 +0100] - ERR - ldbm_config_read_instance_entries -  
Failed to add instance entry cn=userroot,cn=ldbm database,cn=plugins,cn=config  
[04/Feb/2020:13:11:02.471079045 +0100] - ERR - bdb_config_load_dse_info - failed to read  
instance entries  
[04/Feb/2020:13:11:02.476173304 +0100] - ERR - libdb - BDB0522 Page 0: metadata page  
corrupted  
[04/Feb/2020:13:11:02.481684604 +0100] - ERR - libdb - BDB0523 Page 0: could not check  
metadata page  
[04/Feb/2020:13:11:02.484113053 +0100] - ERR - libdb - /var/lib/dirsrv/slapd-  
instance_name/db/userroot/entryrdn.db: BDB0090 DB_VERIFY_BAD: Database verification  
failed  
[04/Feb/2020:13:11:02.486449603 +0100] - ERR - dbverify_ext - verify failed(-30970):  
/var/lib/dirsrv/slapd-instance_name/db/userroot/entryrdn.db  
dbverify failed
```

4. If the verification process reported any problems, fix them manually or restore a backup.

5. Start the Directory Server instance:

```
# dsctl instance_name start
```

# CHAPTER 3. MANAGING DIRECTORY ENTRIES

This chapter discusses how to use command line utilities, such as **ldapmodify**, to manage entries in Directory Server.

To perform LDAP operations using the command line, install the `openldap-clients` package. The utilities installed by this package enable you to:

- Add new entries
- Add new attributes to existing entries
- Update existing entries and attributes
- Delete entries and attributes from entries
- Perform bulk operations

To install the `openldap-clients` package:

```
# yum install openldap-clients
```



## NOTE

To perform LDAP operations, you need the appropriate permissions. For details about access control, see [Chapter 18, Managing Access Control](#).

## 3.1. PROVIDING INPUT TO THE `LDAPADD`, `LDAPMODIFY`, AND `LDAPDELETE` UTILITIES

When you add, update, or delete entries or attributes in your directory, you can either use the interactive mode of the utilities to enter LDAP Data Interchange Format (LDIF) statements or pass an LDIF file to them.

For further details about LDIF, see [Section B.1, “About the LDIF File Format”](#).

### 3.1.1. Providing Input Using the Interactive Mode

In the interactive mode, the **ldapadd**, **ldapmodify**, and **ldapdelete** utilities read the input from the command line. To exit the interactive mode, press the **Ctrl+D (^D)** key combination to send the End Of File (EOF) escape sequence.

In interactive mode, the utility sends the statements to the LDAP server when you press **Enter** twice or when you send the EOF sequence.

Use the interactive mode:

- To enter LDIF statements without creating a file:

#### Example 3.1. Using the `ldapmodify` Interactive Mode to Enter LDIF Statements

The following example starts **ldapmodify** in interactive mode, deletes the **telephoneNumber** attribute, and adds the manager attribute with the **cn=manager\_name,ou=people,dc=example,dc=com** value to the

**uid=user,ou=people,dc=example,dc=com** entry. Press **Ctrl+D** after the last statement to exit the interactive mode.

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user,ou=people,dc=example,dc=com
changetype: modify
delete: telephoneNumber
-
add: manager
manager: cn=manager_name,ou=people,dc=example,dc=com
^D
```

- To redirect LDIF statements, outputted by another command, to Directory Server:

#### Example 3.2. Using the **ldapmodify** Interactive Mode with Redirected Content

The following example redirects the output of the **command\_that\_outputs\_LDIF** command to **ldapmodify**. The interactive mode exits automatically after the redirected command exits.

```
# command_that_outputs_LDIF | ldapmodify -D "cn=Directory Manager" \
-W -p 389 -h server.example.com -x
```

### 3.1.2. Providing Input Using an LDIF File

In the interactive mode, the **ldapadd**, **ldapmodify**, and **ldapdelete** utilities read the LDIF statements from a file. Use this mode to send a larger number of LDIF statements to Directory Server.

#### Example 3.3. Passing a File with LDIF Statements to **ldapmodify**

1. Create a file with the LDIF statements. For example, create the **~/example.ldif** file with the following statements:

```
dn: uid=user,ou=people,dc=example,dc=com
changetype: modify
delete: telephoneNumber
-
add: manager
manager: cn=manager_name,ou=people,dc=example,dc=com
```

This example deletes the **telephoneNumber** attribute and adds the **manager** attribute with the **cn=manager\_name,ou=people,dc=example,dc=com** value to the **uid=user,ou=people,dc=example,dc=com** entry

2. Pass the file to the **ldapmodify** command using the **-f file\_name** option:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
-f ~/example.ldif
```

## 3.2. THE CONTINUOUS OPERATION MODE

If you send multiple LDIF statements to Directory Server and one operation fails, the process stops. However, entries processed before the error occurred were successfully added, modified, or deleted.

To ignore errors and continue processing further LDIF statements in a batch, pass the **-c** option to **ldapadd** and **ldapmodify**. For example:

```
# ldapmodify -c -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

## 3.3. ADDING AN ENTRY

To add a new entry to the directory, use the **ldapadd** or **ldapmodify** utility. Note that **ldapadd** is a symbolic link to **/bin/ldapmodify**. Therefore, **ldapadd** performs the same operation as **ldapmodify -a**.



### NOTE

You can only add a new directory entry, if the parent entry already exists. For example, you cannot add the **cn=user,ou=people,dc=example,dc=com** entry, if the **ou=people,dc=example,dc=com** parent entry does not exist.

### 3.3.1. Adding an Entry Using **ldapadd**

To use the **ldapadd** utility to add, for example, the **cn=user,ou=people,dc=example,dc=com** user entry:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,ou=People,dc=example,dc=com
uid: user
givenName: given_name
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: surname
cn: user
```



### NOTE

Running **ldapadd** automatically performs a **changetype: add** operation. Therefore, you do not need to specify **changetype: add** in the LDIF statement.

For further details on the parameters used in the command, see the **ldapadd(1)** man page.

### 3.3.2. Adding an Entry Using **ldapmodify**

To use the **ldapmodify** utility to add, for example, the **cn=user,ou=people,dc=example,dc=com** user entry:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

```

dn: uid=user,ou=People,dc=example,dc=com
uid: user
givenName: given_name
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: surname
cn: user

```



#### NOTE

When passing the **-a** option to the **ldapmodify** command, the utility automatically performs a **changetype: add** operation. Therefore, you do not need to specify **changetype: add** in the LDIF statement.

For further details on the parameters used in the command, see the **ldapmodify(1)** man page.

### 3.3.3. Creating a Root Entry

To create the root entry of a database suffix, such as **dc=example,dc=com**, bind as the **cn=Directory Manager** user and add the entry.

The DN corresponds to the DN of the root or sub-suffix of the database.

For example, to add the **dc=example,dc=com** suffix:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

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

```



#### NOTE

You can add root objects only if you have one database per suffix. If you create a suffix that is stored in several databases, you must use the **ldif2db** utility with the **-n back\_end** option to set the database that will hold the new entries. For details, see [Section 6.1.2, “Importing Using the Command Line”](#).

## 3.4. UPDATING A DIRECTORY ENTRY

When you modify a directory entry, use the **changetype: modify** statement. Depending on the change operation, you can add, change, or delete attributes from the entry.

Use the **ldapmodify** utility to send the LDIF statements to Directory Server. For example, in interactive mode:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

For further details on the parameters used in **ldapmodify** commands, see the **ldapmodify(1)** man page.

### 3.4.1. Adding Attributes to an Entry

To add an attribute to an entry, use the **add** operation.

For example, to add the **telephoneNumber** attribute with the **555-1234567** value to the **uid=user,dc=people,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
add: telephoneNumber
telephoneNumber: 555-1234567
```

If an attribute is multi-valued, you can specify the attribute name multiple times to add all the values in a single operation. For example, to add two **telephoneNumber** attributes at once to the **uid=user,dc=people,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
add: telephoneNumber
telephoneNumber: 555-1234567
telephoneNumber: 555-7654321
```

### 3.4.2. Updating an Attribute's Value

The procedure for updating an attribute's value depends on if the attribute is single-valued or multi-valued.

#### Updating a Single-value Attribute

When updating a single-value attribute, use the **replace** operation to override the existing value. The following command updates the **manager** attribute of the **uid=user,dc=people,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
replace: manager
manager: uid=manager_name,dc=people,dc=example,dc=com
```

#### Updating a Specific Value of a Multi-value Attribute

To update a specific value of a multi-value attribute, you must first delete the entry you want to replace, and then add the new value. The following command updates only the **telephoneNumber** attribute that is currently set to **555-1234567** in the **uid=user,dc=people,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
```

```

delete: telephoneNumber
telephoneNumber: 555-1234567
-
add: telephoneNumber
telephoneNumber: 555-9876543

```

### 3.4.3. Deleting Attributes from an Entry

To delete an attribute from an entry, use the **delete** operation.

#### Deleting an Attribute

For example, to delete the **manager** attribute from the **uid=user,dc=people,dc=example,dc=com** entry:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
delete: manager

```



#### NOTE

If the attribute contains multiple values, this operation deletes all of them.

#### Deleting a Specific Value of a Multi-value Attribute

If you want to delete a specific value from a multi-value attribute, list the attribute and its value in the LDIF statement. For example, to delete only the **telephoneNumber** attribute that is set to **555-1234567** from the **uid=user,dc=people,dc=example,dc=com** entry:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user,dc=people,dc=example,dc=com
changetype: modify
delete: telephoneNumber
telephoneNumber: 555-1234567

```

## 3.5. DELETING AN ENTRY

Deleting an entry removes the entry from the directory.



#### NOTE

You can only delete entries that have no child entries. For example, you cannot delete the **ou=People,dc=example,dc=com** entry, if the **uid=user,ou=People,dc=example,dc=com** entry still exists.

### 3.5.1. Deleting an Entry Using **ldapdelete**

The **ldapdelete** utility enables you to delete one or multiple entries. For example, to delete the **uid=user,ou=People,dc=example,dc=com** entry:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
"uid=user,ou=People,dc=example,dc=com"
```

To delete multiple entries in one operation, append them to the command. For example:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
"uid=user1,ou=People,dc=example,dc=com" \
"uid=user2,ou=People,dc=example,dc=com"
```

For further details on the parameters used, see the `ldapdelete(1)` man page.

### 3.5.2. Deleting an Entry Using `ldapmodify`

To delete an entry using the `ldapmodify` utility, use the **changetype: delete** operation. For example, to delete the `uid=user,ou=People,dc=example,dc=com` entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user,dc=people,dc=example,dc=com
changetype: delete
```

## 3.6. RENAMING AND MOVING AN ENTRY

This section explains how to rename or move entries.



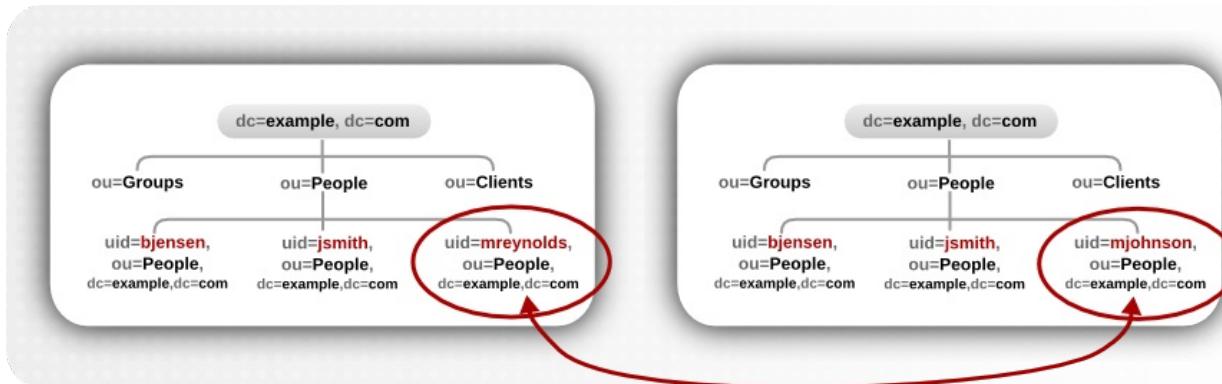
### NOTE

Use the **moddn** Access Control List (ACL) to grant permissions to move entries. For details, see [Section 18.8.2.1, “Targeting Source and Destination DNs”](#).

The following rename operations exist:

#### Renaming an Entry

If you rename a entry, the **modrdn** operation changes the Relative Distinguished Name (RDN) of the entry:



#### Renaming a Subentry

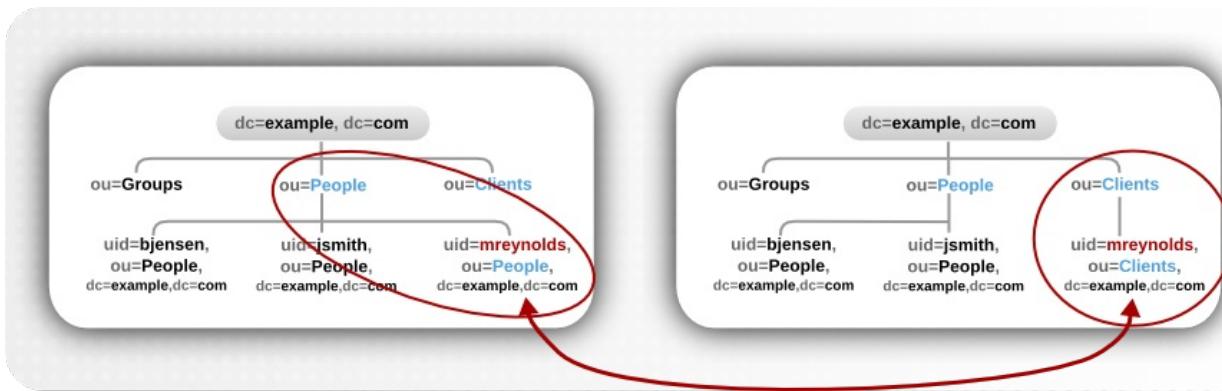
For subtree entries, the **modrdn** operation renames the subtree and also the DN components of child entries:



Note that for large subtrees, this process can take a lot of time and resources.

### Moving an Entry to a New Parent

A similar action to renaming a subtree is moving an entry from one subtree to another. This is an expanded type of the **modrdn** operation, which simultaneously renames the entry and sets a ***newSuperior*** attribute which moves the entry from one parent to another:



#### 3.6.1. Considerations for Renaming Entries

Keep the following in mind when performing rename operations:

- You cannot rename the root suffix.
- Subtree rename operations have minimal effect on replication. Replication agreements are applied to an entire database, not a subtree within the database. Therefore, a subtree rename operation does not require reconfiguring a replication agreement. All name changes after a subtree rename operation are replicated as normal.
- Renaming a subtree might require any synchronization agreements to be reconfigured. Synchronization agreements are set at the suffix or subtree level. Therefore, renaming a subtree might break synchronization.
- Renaming a subtree requires that any subtree-level Access Control Instructions (ACI) set for the subtree be reconfigured manually, as well as any entry-level ACIs set for child entries of the subtree.
- Trying to change the component of a subtree, such as moving from **ou** to **dc**, might fail with a schema violation. For example, the **organizationalUnit** object class requires the **ou** attribute. If that attribute is removed as part of renaming the subtree, the operation fails.
- If you move a group, the **MemberOf** plug-in automatically updates the **memberOf** attributes.

However, if you move a subtree that contain groups, you must manually create a task in the **cn=memberof** task entry or use the **fixup-memberof.pl** to update the related **memberOf** attributes.

For details about cleaning up **memberOf** attribute references, see [Section 8.1.4.8, "Regenerating \*\*memberOf\*\* Values"](#).

### 3.6.2. Renaming Users, Groups, POSIX Groups, and OUs

The **dsidm** utility can rename several types of objects:

- Users:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" user
rename current_user_name new_user_name
```

Note that the **dsidm user rename** command automatically places **ou=People** in front of the base DN you have specified.

- Groups:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
group rename current_group_name new_group_name
```

Note that the **dsidm group rename** command automatically places **ou=Groups** in front of the base DN you have specified.

- POSIX Groups:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
posixgroup rename current_posix_group_name new_posix_group_name
```

Note that the **dsidm posixgroup rename** command automatically places **ou=Groups** in front of the base DN you have specified.

- Organizational Units (OU)

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
organizationalunit rename current_ou_name new_ou_name
```

The **dsidm organizationalunit rename** command performs the rename operation directly in the base DN you have specified.

### 3.6.3. The **deleteOldRDN** Parameter When Renaming Entries Using LDIF Statements

When you rename an entry, the **deleteOldRDN** parameter controls whether the old RDN will be deleted or retained.

#### **deleteOldRDN: 0**

The existing RDN is retained as a value in the new entry. The resulting entry contains two **cn** attributes: one with the old and one with the new common name (CN).

For example, the following attributes belong to a group that was renamed from **cn=old\_group,dc=example,dc=com** to **cn=new\_group,dc=example,dc=com** with the **deleteOldRDN: 0** parameter set.

```
dn: cn=new_group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: old_group
cn: new_group
```

#### **deleteOldRDN: 1**

Directory Server deletes the old entry and creates a new entry using the new RDN. The new entry only contains the **cn** attribute of the new entry.

For example, the following group was renamed to **cn=new\_group,dc=example,dc=com** with the **deleteOldRDN: 1** parameter set:

```
dn: cn=new_group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupofuniqueNames
cn: new_group
```

### 3.6.4. Renaming an Entry or Subtree Using LDIF Statements

To rename an entry or subtree, use the **changetype: modrdn** operation and set the new RDN in the **newrdn** attribute.

For example, to rename the **cn=demo1,dc=example,dc=com** entry to **cn=demo1,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=demo1,dc=example,dc=com
changetype: modrdn
newrdn: cn=demo2
deleteOldRDN: 1
```

For details about the **deleteOldRDN**, see [Section 3.6.3, “The \*\*deleteOldRDN\*\* Parameter When Renaming Entries Using LDIF Statements”](#).

### 3.6.5. Moving an Entry to a New Parent Using LDIF Statements

To move an entry to a new parent, use the **changetype: modrdn** operation and set the following to attributes:

#### **newrdn**

Sets the RDN of the moved entry. You must set this entry, even if the RDN remains the same.

#### **newSuperior**

Sets the DN of the new parent entry.

For example, to move the **cn=demo** entry from **ou=Germany,dc=example,dc=com** to **ou=France,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=demo,ou=Germany,dc=example,dc=com
changetype: modrdn
newrdn: cn=demo
newSuperior= ou=France,dc=example,dc=com
deleteOldRDN: 1
```

For details about the **deleteOldRDN**, see [Section 3.6.3, “The \*\*deleteOldRDN\*\* Parameter When Renaming Entries Using LDIF Statements”](#).

## 3.7. USING SPECIAL CHARACTERS

When using the command line, enclose characters that have a special meaning to the command-line interpreter, such as space ( ), asterisk (\*), or backslash (\), with quotation marks. Depending on the command-line interpreter, use single or double quotation marks.

For example, to authenticate as the **cn=Directory Manager** user, enclose the user's DN in quotation marks:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

Additionally, if a DN contains a comma in a component, escape it using a backslash. For example, to authenticate as the **uid=user,ou=People,dc=example.com Chicago, IL** user:

```
# ldapmodify -a -D "cn=uid=user,ou=People,dc=example.com Chicago\, IL" \
-W -p 389 -h server.example.com -x
```

## 3.8. USING BINARY ATTRIBUTES

Certain attributes support binary values, such as the **jpegPhoto** attribute. When you add or update such an attribute, the utility reads the value for the attribute from a file. To add or update such an attribute, you can use the **ldapmodify** utility.

For example, to add the **jpegPhoto** attribute to the **uid=user,ou=People,dc=example,dc=com** entry, and read the value for the attribute from the **/home/user\_name/photo.jpg** file, enter:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user,ou=People,dc=example,dc=com
changetype: modify
add: jpegPhoto
jpegPhoto:< file:///home/user_name/photo.jpg
```



### IMPORTANT

Note that there is no space between : and <.

## 3.9. UPDATING AN ENTRY IN AN INTERNATIONALIZED DIRECTORY

To use attribute values with languages other than English, associate the attribute's value with a language tag.

When using **ldapmodify** to update an attribute that has a language tag set, you must match the value and language tag exactly or the operation will fail.

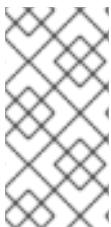
For example, to modify an attribute value that has the **lang-fr** language tag set, include the tag in the **modify** operation:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
  
dn: uid=user,ou=People,dc=example,dc=com  
changetype: modify  
replace: homePostalAddress;lang-fr  
homePostalAddress;lang-fr: 34 rue de Seine
```

# CHAPTER 4. TRACKING MODIFICATIONS TO DIRECTORY ENTRIES

In certain situations it is useful to track when changes are made to entries. There are two aspects of entry modifications that the Directory Server tracks:

- Using change sequence numbers to track changes to the database. This is similar to change sequence numbers used in replication and synchronization. Every normal directory operation triggers a sequence number.
- Assigning creation and modification information. These attributes record the names of the user who created and most recently modified an entry, as well as the timestamps of when it was created and modified.



## NOTE

The entry update sequence number (USN), modify time and name, and create time and name are all operational attributes and are not returned in a regular **ldapsearch**. For details on running a search for operational attributes, see [Section 14.3.7, "Searching for Operational Attributes"](#).

## 4.1. TRACKING MODIFICATIONS TO THE DATABASE THROUGH UPDATE SEQUENCE NUMBERS

The USN Plug-in enables LDAP clients and servers to identify if entries have been changed.

### 4.1.1. An Overview of the Entry Sequence Numbers

When the USN Plug-in is enabled, update sequence numbers (USNs) are sequential numbers that are assigned to an entry whenever a write operation is performed against the entry. (Write operations include add, modify, modrdn, and delete operations. Internal database operations, like export operations, are not counted in the update sequence.) A USN counter keeps track of the most recently assigned USN.

#### 4.1.1.1. Local and Global USNs

The USN is evaluated globally, for the entire database, not for the single entry. The USN is similar to the change sequence number for replication and synchronization, in that it simply ticks upward to track any changes in the database or directory. However, the entry USN is maintained separately from the CSNs, and USNs are not replicated.

The entry shows the change number for the last modification to that entry in the **entryUSN** operational attribute. For further details about operational attributes, see [Section 14.3.7, "Searching for Operational Attributes"](#).

#### Example 4.1. Example Entry USN

To display the **entryusn** attribute of the **uid=example,ou=People,dc=example,dc=com** user entry:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com:389 -x -b
"uid=example,ou=People,dc=example,dc=com" -s base -x entryusn
```

```
dn: uid=example,ou=People,dc=example,dc=com
entryusn: 17653
```

The USN Plug-in has two modes, local mode and global mode:

- In local mode, each back end database has an instance of the USN Plug-in with a USN counter specific to that back end database. This is the default setting.
- In global mode, there is a global instance of the USN Plug-in with a global USN counter that applies to changes made to the entire directory.

When the USN Plug-in is set to local mode, results are limited to the local back end database. When the USN Plug-in is set to global mode, the returned results are for the entire directory.

The root DSE shows the most recent USN assigned to any entry in the database in the ***lastusn*** attribute. When the USN Plug-in is set to local mode, so each database has its own local USN counter, the ***lastUSN*** shows both the database which assigned the USN and the USN:

```
lastusn;database_name:USN
```

For example:

```
lastusn;example1: 2130
lastusn;example2: 2070
```

In global mode, when the database uses a shared USN counter, the ***lastUSN*** attribute shows the latest USN only:

```
lastusn: 4200
```

#### 4.1.1.2. Importing USN Entries

When entries are imported, the USN Plug-in uses the ***nsslapd-entryusn-import-initval*** attribute to check if the entry has an assigned USN. If the value of ***nsslapd-entryusn-import-initval*** is numerical, the imported entry will use this numerical value as the entry's USN. If the value of ***nsslapd-entryusn-import-initval*** is not numerical, the USN Plug-in will use the value of the ***lastUSN*** attribute and increment it by one as the USN for the imported entry.

### 4.1.2. Enabling the USN Plug-in

This section describes how to enable the **USN** plug-in to record USNs on entries.

#### 4.1.2.1. Enabling the USN Plug-in Using the Command Line

To enable the **USN** plug-in using the command line:

1. Use the **dsconf** utility to enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin usn enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 4.1.2.2. Enabling the USN Plug-in Using the Web Console

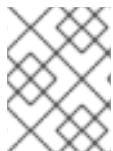
To enable the **USN** plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **USN** plug-in.
5. Change the status to **ON** to enable the plug-in.
6. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 4.1.3. Global USNs

With the default settings, Directory Server uses unique update sequence numbers (USN) for each back end database. Alternatively, you can enable unique USNs across all back end databases.



##### NOTE

The **USN** plug-in must be enabled to use this feature. See [Section 4.1.2, “Enabling the USN Plug-in”](#).

##### 4.1.3.1. Identifying Whether Global USNs are Enabled

This section describes how to identify whether USNs are enabled across all back end databases.

###### 4.1.3.1.1. Identifying Whether Global USNs are Enabled Using the Command Line

To display the current status of the global USN feature using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin usn global
USN global mode is disabled
```

###### 4.1.3.1.2. Identifying Whether Global USNs are Enabled Using the Web Console

To display the current status of the global USN feature using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).

2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **USN** plug-in.
5. Verify that the **USN Global** switch is set to **On**.

#### 4.1.3.2. Enabling Global USNs

##### 4.1.3.2.1. Enabling Global USNs Using the Command Line

To enable global USNs using the command line:

1. Enable global USNs:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin usn global on
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

##### 4.1.3.2.2. Enabling Global USNs Using the Web Console

To enable global USNs using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **USN** plug-in.
5. Change the status of the plug-in to **On**.
6. Change the **USN Global** status to **On**.
7. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 4.1.4. Cleaning up USN Tombstone Entries

The **USN** plug-in moves entries to tombstone entries when the entry is deleted. If replication is enabled, then separate tombstone entries are kept by both the **USN** and **Replication** plug-ins. Both tombstone entries are deleted by the replication process, but for server performance, it can be beneficial to delete the USN tombstones:

- before converting a server to a replica
- to free memory for the server

#### 4.1.4.1. Cleaning up USN Tombstone Entries Using the Command Line

To remove all USN tombstone entries from the **dc=example,dc=com** suffix using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin usn cleanup -s
"dc=example,dc=com"
```

Optionally, pass the **-o max\_USN** option to the command to delete USN tombstone entries up to the specified value.

#### 4.1.4.2. Cleaning up USN Tombstone Entries Using the Web Console

To remove all USN tombstone entries from the **dc=example,dc=com** suffix using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **USN** plug-in.
5. Press the **Run Fixup Task** button.
6. Fill the fields, and press **Run**.

## 4.2. TRACKING ENTRY MODIFICATIONS THROUGH OPERATIONAL ATTRIBUTES

Using the default settings, Directory Server tracks the following operational attributes for every entry:

- **creatorsName**: The distinguished name (DN) of the user who initially created the entry.
- **createTimestamp**: The time stamp in Greenwich Mean Time (GMT) format when the entry was created.
- **modifiersName**: The distinguished name of the user who last modified the entry.
- **modifyTimestamp**: The time stamp in the GMT format for when the entry was last modified.

Note that operational attributes are not returned in default searches. You must explicitly request these attributes in queries. For details, see [Section 14.3.7, “Searching for Operational Attributes”](#).



### IMPORTANT

Red Hat recommends not disabling tracking these operational attributes. If disabled, entries do not get a unique ID assigned in the **nsUniqueId** attribute and replication does not work.

## 4.2.1. Entries Modified or Created by a Database Link

When an entry is created or modified over a database link, the ***creatorsName*** and ***modifiersName*** attributes contain the name of the user who is granted proxy authorization rights on the remote server. In this case, the attributes do not display the original creator or latest modifier of the entry. However, the access logs show both the proxy user (**dn**) and the original user (**authzid**). For example:

```
[23/May/2018:18:13:56.145747965 +051800] conn=1175 op=0 BIND dn="cn=proxy
admin,ou=People,dc=example,dc=com" method=128 version=3
[23/May/2018:18:13:56.575439751 +051800] conn=1175 op=0 RESULT err=0 tag=97 nentries=0
etime=0 dn="cn=proxy admin,ou=people,dc=example,dc=com"
[23/May/2018:18:13:56.744359706 +051800] conn=1175 op=1 SRCH base="dc=example,dc=com"
scope=2 filter="(objectClass=*)" attrs=ALL
authzid="uid=user_name,ou=People,dc=example,dc=com"
```

## 4.2.2. Enabling Tracking of Modifications

By default, Directory Server tracks modifications in operational attributes.



### NOTE

Red Hat recommends not disabling this feature.

This section describes how to re-enable tracking of modifications in case that you disabled the feature.

### 4.2.2.1. Enabling Tracking Of Modifications Using the Command Line

To re-enable tracking of entry modifications using the command line:

1. Set the ***nsslapd-lastmod*** parameter to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
lastmod=on
```

2. Optionally, to regenerate the missing ***nsUniqueId*** attributes:

- a. Export the database into an LDAP Data Interchange Format (LDIF) file. See [Section 6.2.1, “Exporting Data into an LDIF File Using the Command Line”](#).
- b. Import the database from the LDIF file. See [Section 6.1.2, “Importing Using the Command Line”](#).

## 4.3. TRACKING THE BIND DN FOR PLUG-IN INITIATED UPDATES

One change to an entry can trigger other, automatic changes across the directory tree. When a user is deleted, for example, that user is automatically removed from any groups it belonged to by the **Referential Integrity Postoperation** plug-in.

The initial action is shown in the entry as being performed by whatever user account is bound to the server, but all related updates (by default) are shown as being performed by the plug-in, with no information about which user initiated that update. For example, using the MemberOf Plug-in to update user entries with group membership, the update to the group account is shown as being performed by the bound user, while the edit to the user entry is shown as being performed by the MemberOf Plug-in:

```
dn: cn=example_group,ou=groups,dc=example,dc=com
modifiersname: uid=example,ou=people,dc=example,dc=com
```

```
dn: uid=example,ou=people,dc=example,dc=com
modifiersname: cn=memberOf plugin,cn=plugins,cn=config
```

The ***nsslapd-plugin-binddn-tracking*** parameter enables the server to track which user originated an update operation, as well as the internal plug-in which actually performed it. The bound user is shown in the ***modifiersname*** and ***creatorsname*** operational attributes, while the plug-in which performed it is shown in the ***internalModifiersname*** and ***internalCreatorsname*** operational attributes. For example:

```
dn: uid=example,ou=people,dc=example,dc=com
modifiersname: uid=admin,ou=people,dc=example,dc=com
internalModifiersname: cn=memberOf plugin,cn=plugins,cn=config
```

The ***nsslapd-plugin-binddn-tracking*** parameter tracks and maintains the relationship between the bound user and any updates performed for that connection.



#### NOTE

The ***internalModifiersname*** and ***internalCreatorsname*** attributes always show a plug-in as the identity. This plug-in could be an additional plug-in, such as the MemberOf Plug-in. If the change is made by the core Directory Server, then the plug-in is the database plug-in, ***cn=ldbm database,cn=plugins,cn=config***.

### 4.3.1. Enabling Tracking the Bind DN for Plug-in Initiated Updates Using the Command Line

To enable tracking the Bind DN for plug-in-initiated updates using the command line:

1. Set the ***nsslapd-plugin-binddn-tracking*** parameter to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
plugin-binddn-tracking=on
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 4.3.2. Enabling Tracking the Bind DN for Plug-in Initiated Updates Using the Web Console

To enable tracking the Bind DN for plug-in-initiated updates using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the **Server Settings** entry.

4. On the **Advanced Settings** tab, select **Enable Plugin Bind DN Tracking**.
5. Click **Save**.
6. Restart the instance. See [Section 1.5.2, "Starting and Stopping a Directory Server Instance Using the Web Console"](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, "Enabling Plug-ins Dynamically"](#), restarting the instance is not required.

## 4.4. TRACKING PASSWORD CHANGE TIMES

Password change operations are normally treated as any other modification to an entry, so the update time is recorded in the ***lastModified*** operational attribute. However, there can be times when the time of the last password change needs to be recorded separately, to make it easier to update passwords in Active Directory synchronization or to connect with other LDAP clients.

The ***passwordTrackUpdateTime*** attribute within the password policy tells the server to record a timestamp for the last time that the password was updated for an entry. The password change time itself is stored as an operational attribute on the user entry, ***pwdUpdateTime*** (which is separate from the ***modifyTimestamp*** or ***lastModified*** operational attributes).

The ***passwordTrackUpdateTime*** attribute can be set as part of the global password policy or on a subtree or user-level policy, depending on what clients need to access the password change time. Setting password policies is described in [Section 20.4, "Managing the Password Policy"](#).

# CHAPTER 5. MAINTAINING REFERENTIAL INTEGRITY

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

For example, if a user's entry is removed from the directory and Referential Integrity is enabled, the server also removes the user from any groups of which the user is a member. If Referential Integrity is not enabled, the user remains a member of the group until manually removed by the administrator. This is an important feature if you are integrating the Directory Server with other products that rely on the directory for user and group management.

## 5.1. HOW REFERENTIAL INTEGRITY WORKS

When the **Referential Integrity Postoperation** plug-in is enabled, it performs integrity updates on specified attributes immediately after a delete or rename operation. By default, the **Referential Integrity Postoperation** plug-in is disabled.



### NOTE

Enable the **Referential Integrity Postoperation** plug-in only on one supplier replica in a multi-supplier replication environment, because the operations generated by the plug-in will be replicated. If you enable the plug-in on multiple suppliers, the servers have to manage and reapply already performed operations.

When a user or group entry is deleted, updated, renamed, or moved within the directory, the operation is logged to the Referential Integrity log file. For the distinguished names (DN) in the log file, Directory Server searches and updates in intervals the attributes set in the plug-in configuration:

- For entries, marked in the log file as deleted, the corresponding attribute in the directory is deleted.
- For entries, marked in the log file as updated, the corresponding attribute in the directory is updated.
- For entries, marked in the log file as renamed or moved, the value of the corresponding attribute in the directory is renamed.

By default, when the **Referential Integrity Postoperation** plug-in is enabled, it performs integrity updates on the **member**, **uniqueMember**, **owner**, and **seeAlso** attributes immediately after a delete or rename operation. However, the behavior of the **Referential Integrity Postoperation** plug-in can be configured to suit the needs of the directory in several different ways:

- Record Referential Integrity updates in the replication change log.
- Modify the update interval.
- Select the attributes to which to apply Referential Integrity.
- Disable Referential Integrity.

All attributes used in referential integrity *must* be indexed for presence and equality; not indexing those attributes results poor server performance for modify and delete operations.

```
nsIndexType: pres
nsIndexType: eq
nsIndexType: sub
```

See [Section 13.2, “Creating Standard Indexes”](#) for more information about checking and creating indexes.

## 5.2. USING REFERENTIAL INTEGRITY WITH REPLICATION

There are certain limitations when using the **Referential Integrity Postoperation** plug-in in a replication environment:

- Never enable it on a dedicated consumer server (a server that contains only read-only replicas).
- Never enable it on a server that contains a combination of read-write and read-only replicas.
- It is possible to enable it on a supplier server that contains only read-write replicas.
- With multi-supplier replication, enable the plug-in on just one supplier.

If the replication environment satisfies the all of those condition, you can enable the **Referential Integrity Postoperation** plug-in.

1. Enable the **Referential Integrity Postoperation** plug-in as described in [Section 5.3, “Enabling Referential Integrity”](#).
2. Configure the plug-in to record any integrity updates in the changelog.
3. Ensure that the **Referential Integrity Postoperation** plug-in is disabled on all consumer servers.



### NOTE

Because the supplier server sends any changes made by the **Referential Integrity Postoperation** plug-in to consumer servers, it is unnecessary to run the **Referential Integrity Postoperation** plug-in on consumer servers.

## 5.3. ENABLING REFERENTIAL INTEGRITY

This section describes how to enable the **Referential Integrity Postoperation** plug-in.

### 5.3.1. Enabling Referential Integrity Using the Command Line

To enable the **Referential Integrity Postoperation** plug-in using the command line:

1. Use the **dsconf** utility to enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity
enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 5.3.2. Enabling Referential Integrity Using the Web Console

To enable the **Referential Integrity** plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **Referential Integrity** plug-in and click **Show Advanced Settings**.
5. Change the status to **ON** to enable the plug-in.

## 5.4. THE REFERENTIAL INTEGRITY UPDATE INTERVAL

By default, the server performs Referential Integrity updates immediately after a **delete** or a **modrdn** operation. Depending on the amount of operations, this can cause a performance impact. To reduce the performance impact, you can increase the amount of time between updates.

You can set the update interval in seconds. Alternatively, you can set the following values:

- **0**: The check for referential integrity is performed immediately.
- **-1**: No check for referential integrity is performed.



#### IMPORTANT

If you set the update interval to **0**, you can only enable the plug-in on all suppliers in a multi-supplier replication environment if you also set their **Referential Integrity Postoperation** plug-in's update interval to **0**. However, if you configure a positive value on one supplier, you must not enable the plug-in on any other supplier to prevent replication loops and directory inconsistencies.

If you want to enable the plug-in in a multi-supplier replication environment, Red Hat recommends setting the update interval to **0** and to enable the plug-in on all suppliers.



#### NOTE

**Referential Integrity** can only be enabled on one supplier. If you set the interval to **0**, Directory Server cleans up references replicates these changes to all consumers immediately. If you set the interval to a value greater than **0**, and the supplier who has **Referential Integrity** enabled is offline, the references are not cleaned up before this supplier is up again.

### 5.4.1. Displaying the Update Interval Using the Command Line

To display the update interval using the command line to:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity show  
referint-update-delay: 0  
...
```

#### 5.4.2. Displaying the Update Interval Using the Web Console

To display the update interval using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. See the **Update Delay** field for the update interval.

#### 5.4.3. Modifying the Update Interval Using the Command Line

To set the update interval using the command line to, for example, to update immediately:

1. Set the update interval to **0**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set  
--update-delay=0
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 5.4.4. Modifying the Update Interval Using the Web Console

To set the update interval using the web console, for example, to update immediately:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. Set the interval in the **Update Delay** field.
6. Press **Save Config**.
7. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

## 5.5. DISPLAYING AND 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 using the command line or the web console.



### NOTE

Attributes set in the **Referential Integrity** plug-in's parameter list, must have equality indexing on all databases. Otherwise, the plug-in scans every entry of the database for matching the deleted or modified DN. This can have a significant performance impact. For details about checking and creating indexes, see [Section 13.2, “Creating Standard Indexes”](#).

### 5.5.1. Displaying the Attribute List Using the Command Line

To display the attribute list using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity show
```

### 5.5.2. Displaying the Attribute List Using the Web Console

To display the attribute list using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. See the **Membership Attribute** field for the list of attributes.

### 5.5.3. Configuring the Attribute List Using the Command Line

To update the attribute list using the command line:

1. Optionally, display the current list of attributes. See [Section 5.5.1, “Displaying the Attribute List Using the Command Line”](#).
2. Update the attribute list:
  - To set an attribute list that should be checked and updated by the plug-in:
 

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --membership-attr attribute_name_1 attribute_name_2
```
  - To delete all attributes that should no longer be checked and updated by the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity
set --membership-attr delete
```

3. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 5.5.4. Configuring the Attribute List Using the Web Console

To update the attribute list using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. Update the **Membership Attribute** field to set the attributes.
  - To add an attribute, enter the name into the **Membership Attribute** field.
  - To remove an attribute, press the **X** button right next to the attribute's name in the **Membership Attribute** field.
6. Press **Save Config**.

## 5.6. CONFIGURING SCOPE FOR THE REFERENTIAL INTEGRITY

If an entry is deleted, the references to it are deleted or modified to reflect the change. When this update is applied to all entries and all groups, it can impact performance and prevents flexibility of restricting the referential integrity to selected subtrees. Defining a scope addresses this problem.

For example, there may be one suffix, **dc=example,dc=com**, containing two subtrees: **ou=active users,dc=example,dc=com** and **ou=deleted users,dc=example,dc=com**. Entries in **deleted users** should not be handled for purposes of referential integrity.

### 5.6.1. Parameters That Control the Referential Integrity Scope

The following three parameters can be used to define the scope in the **Referential Integrity Postoperation** plug-in configuration:

#### ***nsslapd-pluginEntryScope***

This multi-value parameter controls the scope of the entry that is deleted or renamed. It defines the subtree in which the **Referential Integrity Postoperation** plug-in looks for the delete or rename operations of a user entry. If a user is deleted or renamed that does not exist under the defined subtree, the plug-in ignores the operation. The parameter allows you to specify to which branches of the database the plug-in should apply the operation.

### ***nsslapd-pluginExcludeEntryScope***

This parameter also controls the scope of the entry that is deleted or renamed. It defines the subtree in which the **Referential Integrity Postoperation** plug-in ignores any operations for deleting or renaming a user.

### ***nsslapd-pluginContainerScope***

This parameter controls the scope of groups in which references are updated. After a user is deleted, the **Referential Integrity Postoperation** plug-in looks for the groups to which the user belongs and updates them accordingly. This parameter specifies which branch the plug-in searches for the groups to which the user belongs. The **Referential Integrity Postoperation** plug-in only updates groups that are under the specified container branch, and leaves all other groups not updated.

## 5.6.2. Displaying the Referential Integrity Scope Using the Command Line

The following commands show how to display the scope settings using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity show
...
nsslapd-pluginEntryScope: DN
nsslapd-pluginExcludeEntryScope: DN
nsslapd-pluginContainerScope: DN
```

## 5.6.3. Displaying the Referential Integrity Scope Using the Web Console

The following procedure shows how to display the scope settings using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. See the **Entry Scope**, **Exclude Entry Scope**, and **Container Scope** fields for the currently configured scope.

## 5.6.4. Configuring the Referential Integrity Scope Using the Command Line

To configure the referential integrity scope using the command line:

1. Optionally, display the scope settings. See [Section 5.6.2, “Displaying the Referential Integrity Scope Using the Command Line”](#).
2. The following commands show how to configure the individual referential integrity scope settings using the command line:
  - To set a distinguished name (DN):
    - To the ***nsslapd-pluginEntryScope*** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --entry-scope="DN"
```

- To the **nsslapd-pluginExcludeEntryScope** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --exclude-entry-scope="DN"
```

- To the **nsslapd-pluginContainerScope** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --container-scope="DN"
```

- To remove a DN:

- From the **nsslapd-pluginEntryScope** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --entry-scope=delete
```

- From the **nsslapd-pluginExcludeEntryScope** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --exclude-entry-scope=delete
```

- From the **nsslapd-pluginContainerScope** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --container-scope=delete
```

3. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

## 5.6.5. Configuring the Referential Integrity Scope Using the Web Console

To configure the referential integrity scope using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **Referential Integrity** plug-in.
5. Set the scope in the **Entry Scope**, **Exclude Entry Scope**, and **Container Scope** fields.

6. Click **Save Config**.

# CHAPTER 6. POPULATING DIRECTORY DATABASES

Databases contain the directory data managed by Red Hat Directory Server.

## 6.1. IMPORTING DATA

Directory Server can populate a database with data by:

- Importing data
- Initializing a database for replication

[Table 6.1, “Import Method Comparison”](#) describes the differences between an import and initializing databases.

**Table 6.1. Import Method Comparison**

Action	Import	Initialize Database
Overwrites database	No	Yes
LDAP operations	Add, modify, delete	Add only
Performance	More time-consuming	Fast
Partition specialty	Works on all partitions	Local partitions only
Response to server failure	Best effort (all changes made up to the point of the failure remain)	Atomic (all changes are lost after a failure)
LDIF file location	Local to the web console	Local to the web console or local to server
Imports configuration information <b>(cn=config)</b>	Yes	No

### 6.1.1. Setting EntryUSN Initial Values During Import

Entry update sequence numbers (USNs) are not preserved when entries are exported from one server and imported into another. As [Section 4.1, “Tracking Modifications to the Database through Update Sequence Numbers”](#) explains, entry USNs are assigned for operations that happen on a local server, so it does not make sense to import those USNs onto another server.

However, it is possible to configure an initial entry USN value for entries when importing a database or initializing a database (such as when a replica is initialized for replication). This is done by setting the ***nsslapd-entryusn-import-initval*** parameter, which sets a starting USN for all imported entries.

There are two possible values for ***nsslapd-entryusn-import-initval***:

- An integer, which is the explicit start number used for every imported entry.

- **next**, which means that every imported entry uses whatever the highest entry USN value was on the server before the import operation, incremented by one.

If ***nsslapd-entryusn-import-initval*** is not set, then all entry USNs begin at zero.

#### Example 6.1. How the ***nsslapd-entryusn-import-initval*** Parameter works

For example, if the highest value on the server is **1000** before the import or initialization operation, and the ***nsslapd-entryusn-import-initval*** value is **next**, then every imported entry is assigned a USN of **1001**:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x "(cn=*)" entryusn

dn: dc=example,dc=com
entryusn: 1001
dn: ou=Accounting,dc=example,dc=com
entryusn: 1001
dn: ou=Product Development,dc=example,dc=com
entryusn: 1001
...
dn: uid=user_name,ou=people,dc=example,dc=com
entryusn: 1001
...
```

To set an initial value for entry USNs, add the ***nsslapd-entryusn-import-initval*** parameter to the server into which data are being imported or to the supplier server which will perform the initialization. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-entryusn-
import-initval=next
```



#### NOTE

In multi-supplier replication, the ***nsslapd-entryusn-import-initval*** parameter is *not* replicated between servers. This means that the value must be set specifically on whichever supplier server is being used to initialize a replica.

For example, if the **Supplier1** host has ***nsslapd-entryusn-import-initval*** set to *next* and is used to initialize a replica, then the entry USNs for imported entries have the highest value plus one. If the **Supplier2** host does not have ***nsslapd-entryusn-import-initval*** set and is used to initialize a replica, then all entry USNs for imported entries begin at zero – even if **Supplier1** and **Supplier2** have a multi-supplier replication agreement between them.

### 6.1.2. Importing Using the Command Line

Directory Server supports importing data while the instance is running or while the instance is offline:

- Use one of the following methods if the instance is running:
  - Use the **dsconf backend import** command. See [Section 6.1.2.1.1, “Importing Using the \*\*dsconf backend import\*\* Command”](#).

- Create a **cn=tasks** entry. See [Section 6.1.2.1.2, “Importing Data Using a \*\*cn=tasks\*\* Entry”](#).
- If the instance is offline, use the **dsctl ldif2db** command. See [Section 6.1.2.2, “Importing Data While the Server is Offline”](#).



### WARNING

When you start an import operation, Directory Server first removes all existing data from the database and subsequently imports the data from the LDIF file. If the import fails, for example, because the LDIF file does not exist, the server has already removed the previous data from the database.

Note that the LDIF files used for import operations must use **UTF-8** character set encoding. Import operations do not convert data from the local character set encoding to **UTF-8**. Additionally, all imported LDIF files must contain the root suffix entry.

Directory Server runs import operations as the **dirsrv** user. Therefore, the permissions of the LDIF file must allow this user to read the file.

#### 6.1.2.1. Importing Data While the Server is Running

This section describes how you can import data while Directory Server is running.

##### 6.1.2.1.1. Importing Using the **dsconf backend import** Command

Use the **dsconf backend import** command to automatically create a task that imports data from an LDIF file. For example, to import the `/var/lib/dirsrv/slapd-instance_name/ldif/instance_name-database_name-time_stamp.ldif` file into the **userRoot** database:

1. Create the suffix if it does not exist. For details, see [Section 2.1.1, “Creating Suffixes”](#).
2. If the LDIF you want to import does not contain statements that add the suffix entry, create this entry manually as described in [Section 3.3.3, “Creating a Root Entry”](#).
3. Import the LDIF file:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend import userRoot
/var/lib/dirsrv/slapd-instance_name/ldif/instance_name-database_name-time_stamp.ldif
The import task has finished successfully
```

The **dsconf backend import** command supports additional options, for example, to exclude a specific suffix. To display all available options, enter:

```
# dsconf ldap://server.example.com backend import --help
```

##### 6.1.2.1.2. Importing Data Using a **cn=tasks** Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate an import operation, create a task in the **cn=import,cn=tasks,cn=config** entry.

An import task entry requires the following attributes:

- **cn**: Sets the unique name of the task.
- **nsFilename**: Sets the name of the LDIF file to import.
- **nsInstance**: Sets the name of the database into which the file should be imported.

Import tasks support additional parameters, for example, to exclude suffixes. For a complete list, see the **cn=import** section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to add a task that imports the content of the **/tmp/example.ldif** file into the **userRoot** database:

1. Create the suffix if it does not exist. For details, see [Section 2.1.1, “Creating Suffixes”](#).
2. If the LDIF you want to import does not contain statements that add the suffix entry, create this entry manually as described in [Section 3.3.3, “Creating a Root Entry”](#).
3. Add the import task:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=example_import,cn=import,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example_import
nsFilename: /tmp/example.ldif
nsInstance: userRoot
```

When the task is completed, the entry is removed from the directory configuration.

### 6.1.2.2. Importing Data While the Server is Offline

If the server is offline when you import data, use the **dsctl ldif2db** command:

1. Create the suffix if it does not exist. For details, see [Section 2.1.1, “Creating Suffixes”](#).
2. If the LDIF you want to import does not contain statements that add the suffix entry, create this entry manually as described in [Section 3.3.3, “Creating a Root Entry”](#).
3. Stop the instance:

```
# dsctl instance_name stop
```

4. Import the data from the LDIF file. For example, to import the **/tmp/example.ldif** file into the **userRoot** database:

```
# dsctl instance_name ldif2db userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
[17/Jul/2018:13:42:42.015554231 +0200] - INFO - ldbm_instance_config_cachememsize_set
```

```
- force a minimal value 512000
...
[17/Jul/2018:13:42:44.302630629 +0200] - INFO - import_main_offline - import userroot:
Import complete. Processed 160 entries in 2 seconds. (80.00 entries/sec)
ldif2db successful
```



### WARNING

If the database specified in the command does not correspond with the suffix contained in the LDIF file, all data contained in the database is deleted, and the import fails.

5. Start the instance:

```
# dsctl instance_name start
```

#### 6.1.3. Importing Data Using the Web Console

To import data from an LDIF file using the web console:

1. Create the suffix if it does not exist. For details, see [Section 2.1.1, “Creating Suffixes”](#).
2. If the LDIF you want to import does not contain statements that add the suffix entry, create this entry manually as described in [Section 3.3.3, “Creating a Root Entry”](#).
3. Store the LDIF file you want to import in the `/var/lib/dirsrv/slapd-instance_name/ldif` directory.
4. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
5. Select the instance.
6. Open the **Database** menu.
7. Select the suffix entry.
8. Click **Suffix Tasks**, and select **Initialize Suffix**.
9. Select the LDIF file to import or enter the full path to the file.

**Initialize Database via LDIF File**

LDIF File	Creation Date	Size	
Example.ldif	2020-02-06 13:34:07	43.8M	<b>Import</b>

6 per page 1-1 of 1 << < 1 of 1 > >>

Or, enter LDIF location  **Import**

**Close**

10. Select **Yes, I am sure.**, and click **Initialize Database** to confirm.

## 6.2. EXPORTING DATA

LDAP Data Interchange Format (LDIF) files are used to export database entries from the Directory Server databases. LDIF is a standard format described in [RFC 2849](#).



### NOTE

The export operations do not export the configuration information (**cn=config**), schema information (**cn=schema**), or monitoring information (**cn=monitor**).

Exporting data can be useful for the following:

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

For example, if a directory contains one database, and its contents should be split into two databases, then the two new databases receive their data by exporting the contents of the old databases and importing it into the two new databases, as illustrated in [Figure 6.1, “Splitting a Database Contents into Two Databases”](#).

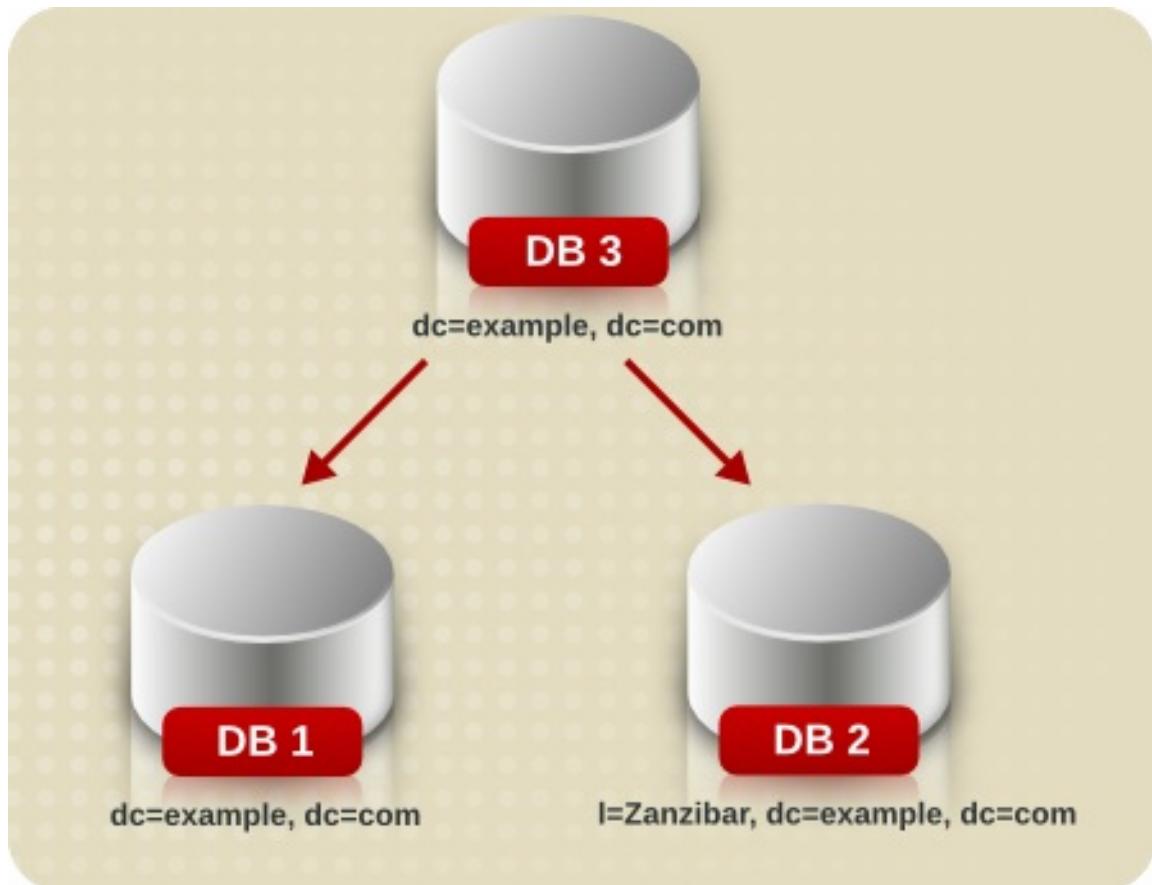


Figure 6.1. Splitting a Database Contents into Two Databases



#### WARNING

Do not stop the server during an export operation.

Directory Server runs the export operations as the **dirsrv** user. Therefore, the permissions of the destination directory must allow this user to write the file.

### 6.2.1. Exporting Data into an LDIF File Using the Command Line

Directory Server supports exporting data while the instance is running or while the instance is offline:

- Use one of the following methods if the instance is running:
  - Use the **dsconf backend export** command. See [Section 6.2.1.1, “Exporting a Databases Using the \*\*dsconf backend export\*\* Command”](#).
  - Create a **cn=tasks** entry. See [Section 6.2.1.2, “Exporting a Database Using a \*\*cn=tasks\*\* Entry”](#).
- If the instance is offline, use the **dsctl db2ldif** command. See [Section 6.2.1.2, “Exporting a Database While the Server is Offline”](#).

## 6.2.1.1. Exporting a Database While the Server is Running

### 6.2.1.1.1. Exporting a Databases Using `dsconf backend export` Command

Use the **`dsconf backend export`** command to automatically create a task that exports data to an LDIF file.

For example, to export the **userRoot** database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export userRoot
The export task has finished successfully
```

By default, **`dsconf`** stores the export in a file called **`instance_name_database_name-time_stamp.ldif`** in the **/var/lib/dirsrv/slappd-instance\_name/export** directory. Alternatively, add the **-I file\_name** option to the command to specify a different location.

The **`dsconf backend export`** command supports additional options, for example, to exclude a specific suffix. To display all available options, enter:

```
# dsconf ldap://server.example.com backend export --help
```

## 6.2.1.1.2. Exporting a Database Using `acn=tasks` Entry

The **`cn=tasks,cn=config`** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate an export operation, create a task in the **`cn=export,cn=tasks,cn=config`** entry.

Using a task entry enables you to export data while the server is running.

An export task entry requires the following attributes:

- **`cn`**: Sets the unique name of the task.
- **`nsInstance`**: Sets the name of the database to export.
- **`nsFilename`**: Sets the name of the file into which the export should be stored.

Export tasks support additional parameters, for example, to exclude suffixes. For a complete list, see the **`cn=export`** section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to add a task that exports the content of the **userRoot** database into the **/tmp/example.ldif** file:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=example_export,cn=export,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example_export
nsInstance: userRoot
nsFilename: /tmp/example.ldif
```

When the task is completed, the entry is removed from the directory configuration.

### 6.2.1.2. Exporting a Database While the Server is Offline

If the server is offline when you export data, use the **dsctl db2ldif** command:

1. Stop the instance:

```
# dsctl instance_name stop
```

2. Export the database into an LDIF file. For example to export the **userRoot** database into the **/tmp/example.ldif** file:

```
# dsctl instance_name db2ldif userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
ldiffile: /tmp/example.ldif
[18/Jul/2018:10:46:03.353656777 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
[18/Jul/2018:10:46:03.383101305 +0200] - INFO - ldbm_back_ldbm2ldif - export userroot:
Processed 160 entries (100%).
[18/Jul/2018:10:46:03.391553963 +0200] - INFO - dblayer_pre_close - All database threads
now stopped
db2ldif successful
```

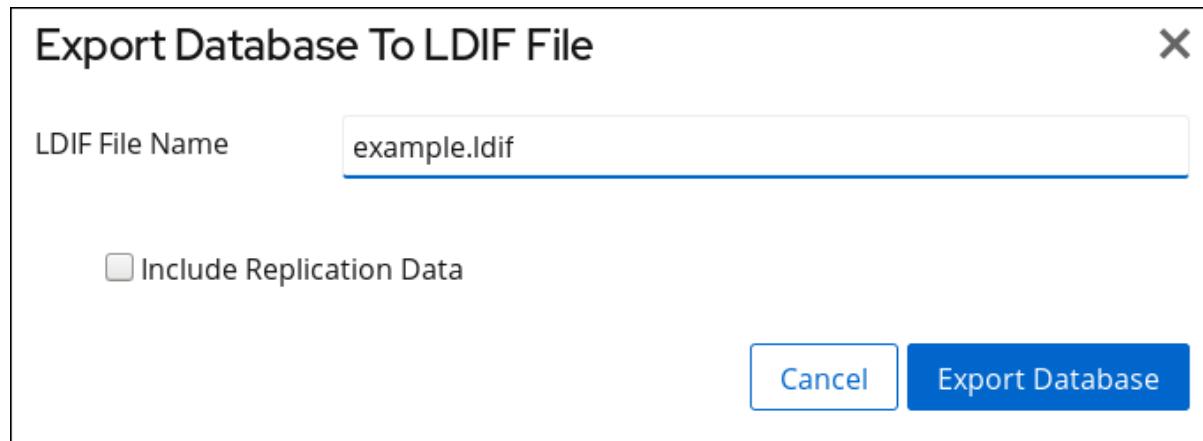
3. Start the instance:

```
# dsctl instance_name start
```

### 6.2.2. Exporting a Suffix to an LDIF File Using the Web Console

To export a suffix using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the suffix entry.
5. Click **Suffix Tasks**, and select **Export Suffix**.
6. Enter the name of the LDIF file in which you want to store the export. Directory Server will store the file in the **/var/lib/dirsrv/slappd-*instance\_name*/ldif/** directory using the specified file name.



7. Click **Export Database**.

### 6.2.3. Enabling Members of a Group to Export Data and Performing the Export as One of the Group Members

You can configure that members of a group have permissions to export data. This increases the security because you no longer need to set the credentials of **cn=Directory Manager** in your scripts. Additionally, you can easily grant and revoke the export permissions by modifying the group.

#### 6.2.3.1. Enabling a Group to Export Data

Use this procedure to add the **cn=export\_users,ou=groups,dc=example,dc=com** group and enable members of this group to create export tasks.

##### Procedure

1. Create the **cn=export\_users,ou=groups,dc=example,dc=com** group:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
group create --cn export_users
```

2. Add access control instructions (ACI) that allows members of the **cn=export\_users,ou=groups,dc=example,dc=com** group to create export tasks:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com
dn: cn=config
changetype: modify
add: aci
aci: (target = "ldap:///cn=export,cn=tasks,cn=config")(targetattr="*")
(version 3.0 ; acl "permission: Allow export_users
group to export data" ; allow (add, read, search) groupdn
= "ldap:///cn=export_users,ou=groups,dc=example,dc=com";)
-
add: aci
aci: (target = "ldap:///cn=config")(targetattr =
"objectclass || cn || nsslapd-suffix || nsslapd-ldifdir")
(version 3.0 ; acl "permission: Allow export_users
group to access ldifdir attribute" ; allow
(read,search) groupdn = "ldap:///cn=export_users,ou=groups,dc=example,dc=com";)
```

3. Create a user:

a. Create a user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
user create --uid="example" --cn="example" --uidNumber="1000" --gidNumber="1000" --
homeDirectory="/home/example/" --displayName="Example User"
```

b. Set a password on the user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account reset_password "uid=example,ou=People,dc=example,dc=com" "password"
```

4. Add the **uid=example,ou=People,dc=example,dc=com** user to the **cn=export\_users,ou=groups,dc=example,dc=com** group:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
group add_member export_users uid=example,ou=People,dc=example,dc=com
```

## Verification

- Display the ACIs set on the **cn=config**:

```
# ldapsearch -o ldif-wrap=no -LLLx -D "cn=Directory Manager" -W -H
ldap://server.example.com -b cn=config aci=* aci -s base
dn: cn=config
aci: (target = "ldap:///cn=export,cn=tasks,cn=config")(targetattr="*")(version 3.0 ; acl
"permission: Allow export_users group to export data" ; allow (add, read, search) groupdn =
"ldap:///cn=export_users,ou=groups,dc=example,dc=com";)
aci: (target = "ldap:///cn=config")(targetattr = "objectclass || cn || nsslapd-suffix || nsslapd-
ldifdir")(version 3.0 ; acl "permission: Allow export_users group to access ldifdir attribute" ;
allow (read,search) groupdn = "ldap:///cn=export_users,ou=groups,dc=example,dc=com";)
...
...
```

### 6.2.3.2. Performing an Export as a Regular User

You can perform exports as a regular user instead of **cn=Directory Manager**.

#### Prerequisites

- You enabled members of the **cn=export\_users,ou=groups,dc=example,dc=com** group to export data. See [Section 6.2.3.1, “Enabling a Group to Export Data”](#).
- The user you use to perform the export is a member of the **cn=export\_users,ou=groups,dc=example,dc=com** group.

#### Procedure

- Create a export task using one of the following methods:

- Using the **dsconf backend export** command:

```
# dsconf -D "uid=example,ou=People,dc=example,dc=com" ldap://server.example.com
backend export userRoot
```

- By manually creating the task:

```
# ldapadd -D "uid=example,ou=People,dc=example,dc=com" -W -H
ldap://server.example.com

dn: cn=userRoot-2021_07_23_12:55_00,cn=export,cn=tasks,cn=config
changetype: add
objectClass: extensibleObject
nsFilename: /var/lib/dirsrv/slapd-instance_name/ldif/None-userroot-
2021_07_23_12:55_00.ldif
nsInstance: userRoot
cn: export-2021_07_23_12:55_00
```

## Verification

- Verify that the backup was created:

```
# ls -l /var/lib/dirsrv/slapd-instance_name/ldif/*.ldif
total 0
-rw----- 1 dirsrv dirsrv 10306 Jul 23 12:55 None-userroot-2021_07_23_12_55_00.ldif
...
```

## 6.3. BACKING UP DIRECTORY SERVER

A backup in Directory Server contains, for example:

- All database files including the data stored within these databases



### NOTE

Directory Server does not support backing up individual databases.

- The transaction logs
- The Indices

In contrast to a backup, you can export data as described in [Section 6.2, “Exporting Data”](#). Use the export feature to export specific data, such as a subtree, from a server in the LDAP Data Interchange Format (LDIF) format.



### WARNING

Do not stop the server during a backup operation.

Directory Server runs the backup task as the **dirsrv** user. Therefore, the permissions of the destination directory must allow this user to create files.

### 6.3.1. Backing up All Databases Using the Command Line

Directory Server supports backing up the databases while the instance is running or while the instance is offline:

- Use one of the following methods if the instance is running:
  - Use the **dsconf backup create** command. See [Section 6.3.1.1, “Backing up All Databases Using the dsconf backup create Command”](#).
  - Create a **cn=tasks** entry. See [Section 6.3.1.2, “Backing up All Databases Using a cn=tasks entry”](#).
- If the instance is offline, use the **dsctl db2bak** command. See [Section 6.3.1.2, “Backing up All Databases While the Server is Offline”](#).



## IMPORTANT

These methods only back up the databases. For details about backing up other important files, such as the configuration, see [Section 6.3.3, “Backing up Configuration Files, the Certificate Database, and Custom Schema Files”](#).

### 6.3.1.1. Backing up All Databases While the Server is Running

#### 6.3.1.1.1. Backing up All Databases Using the **dsconf backup create** Command

Use the **dsconf backup create** command to automatically create a task that backs up all databases.

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backup create
The backup create task has finished successfully
```

By default, **dsconf** stores the backup in a subdirectory called ***instance\_name-time\_stamp*** in the **/var/lib/dirsrv/slapd-*instance\_name*/bak/** directory. To specify a different location, append a directory name to the command.

#### 6.3.1.1.2. Backing up All Databases Using **acn=tasks** entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate a backup operation, create a task in the **cn=backup,cn=tasks,cn=config** entry.

Using a task entry enables you to backup the databases while the server is running.

A backup task entry requires the following attributes:

- **cn**: Sets the unique name of the task.
- **nsDatabaseType**: Sets the type of the database to back up. Directory Server supports only the **ldbm database** value in this attribute.

Backup tasks support additional parameters, for example, to specify a different destination directory as the default, **/var/lib/dirsrv/slapd-*instance\_name*/bak/**. For a complete list, see the **cn=backup** section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to backup all databases and store the archive in the default backup directory:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com -x
```

```
dn: cn=example_backup,cn=export,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example_backup
nsDatabaseType: ldbm database
```

If you not specify the ***nsArchiveDir*** attribute, the server stores the backup in a subdirectory called ***instance\_name-time\_stamp*** in the **/var/lib/dirsrv/slapd-*instance\_name*/bak** directory.

When the task is completed, the entry is removed from the directory configuration.

### 6.3.1.2. Backing up All Databases While the Server is Offline

If the server is offline when you backup databases, use the **dsctl db2bak** command:

1. Stop the instance:

```
# dsctl instance_name stop
```

2. Backup the database:

```
# dsctl instance_name db2bak
db2bak successful
```



#### NOTE

The **dsctl db2bak** command runs as the backup as the **dirsrv** user. Therefore, the permissions of the destination directory must allow this user to create files and directories.

If you not append a destination directory to the command, the server stores the backup in a subdirectory called ***instance\_name-time\_stamp*** in the **/var/lib/dirsrv/slapd-*instance\_name*/bak** directory.

3. Start the instance:

```
# dsctl instance_name start
```

### 6.3.2. Backup up all Databases Using the Web Console

To back up all databases of an instance using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Click the **Actions** button, and select **Manage Backup**.
4. Click **Create Backup**.

5. Enter a name for the backup, such as a time stamp to indicate the creation date and time of the backup.
6. Click **Create Backup**.

The server stores the backup in a subdirectory with the name you entered in the `/var/lib/dirsrv/slappd-instance_name/bak/` directory.

### 6.3.3. Backing up Configuration Files, the Certificate Database, and Custom Schema Files

The backup mechanism integrated into Directory Server backs up only the databases. However, there are additional files stored in the `/etc/dirsrv/slappd-instance_name/` directory which are required to, for example, restore a instance on a different server after a hardware failure.



#### NOTE

Backing up the configuration directory is not supported in the web console.

#### Example 6.2. How to Back up the `/etc/dirsrv/slappd-instance_name/` Directory

To back up the content of `/etc/dirsrv/slappd-instance_name/`, you can copy the directory or store it into an archive file. For example, to store the content of the `/etc/dirsrv/slappd-instance_name/` directory in the `/root/config_slappd-instance_name_time_stamp.tar.gz` file:

```
# cd /etc/dirsrv/
# tar -zcvf /root/config_slappd-instance_name_${(date +%Y-%m-%d_%H-%M-%S)}.tar.gz slappd-instance_name/
```



#### IMPORTANT

During the backup, do not update the certificate database. Otherwise, this database might not be consistent in the backup.

### 6.3.4. Enabling Members of a Group to Back up Directory Server and Performing the Backup as One of the Group Members

You can configure that members of a group have permissions to back up an instance and perform the backup. This increases the security because you no longer need to set the credentials of **cn=Directory Manager** in your backup script or cron jobs. Additionally, you can easily grant and revoke the backup permissions by modifying the group.

#### 6.3.4.1. Enabling a Group to Back up Directory Server

Use this procedure to add the `cn=backup_users,ou=groups,dc=example,dc=com` group and enable members of this group to create backup tasks.

#### Procedure

1. Create the `cn=backup_users,ou=groups,dc=example,dc=com` group:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
group create --cn backup_users
```

2. Add access control instructions (ACI) that allows members of the **cn=backup\_users,ou=groups,dc=example,dc=com** group to create backup tasks:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com

dn: cn=config
changetype: modify
add: aci
aci: (target = "ldap:///cn=backup,cn=tasks,cn=config")(targetattr="*")
(version 3.0 ; acl "permission: Allow backup_users
group to create backup tasks" ; allow (add, read, search) groupdn
= "ldap:///cn=backup_users,ou=groups,dc=example,dc=com";)

-
add: aci
aci: (target = "ldap:///cn=config")(targetattr = "nsslapd-bakdir || objectClass")
(version 3.0 ; acl "permission: Allow backup_users group
to access bakdir attribute" ; allow (read,search) groupdn
= "ldap:///cn=backup_users,ou=groups,dc=example,dc=com";)
```

3. Create a user:

- a. Create a user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
user create --uid="example" --cn="example" --uidNumber="1000" --gidNumber="1000" --
homeDirectory="/home/example" --displayName="Example User"
```

- b. Set a password on the user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account reset_password "uid=example,ou=People,dc=example,dc=com" "password"
```

4. Add the **uid=example,ou=People,dc=example,dc=com** user to the **cn=backup\_users,ou=groups,dc=example,dc=com** group:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
group add_member backup_users uid=example,ou=People,dc=example,dc=com
```

## Verification

- Display the ACIs set on the **cn=config** entry:

```
# ldapsearch -o ldif-wrap=no -LLLx -D "cn=directory manager" -W -H
ldap://server.example.com -b cn=config aci=* aci -s base
dn: cn=config
aci: (target = "ldap:///cn=backup,cn=tasks,cn=config")(targetattr="*")(version 3.0 ; acl
"permission: Allow backup_users group to create backup tasks" ; allow (add, read, search) groupdn
= "ldap:///cn=backup_users,ou=groups,dc=example,dc=com");
aci: (target = "ldap:///cn=config")(targetattr = "nsslapd-bakdir || objectClass")(version 3.0 ; acl
```

```
"permission: Allow backup_users group to access bakdir attribute" ; allow (read,search)
groupdn = "ldap:///cn=backup_users,ou=groups,dc=example,dc=com";)
...

```

### 6.3.4.2. Performing a Backup as a Regular User

You can perform backups as a regular user instead of **cn=Directory Manager**.

#### Prerequisites

- You enabled members of the **cn=backup\_users,ou=groups,dc=example,dc=com** group to perform backups. See [Section 6.3.4.1, “Enabling a Group to Back up Directory Server”](#).
- The user you use to perform the backup is a member of the **cn=backup\_users,ou=groups,dc=example,dc=com** group.

#### Procedure

- Create a backup task using one of the following methods:

- Using the **dsconf backup create** command:

```
# dsconf -D uid=example,ou=People,dc=example,dc=com ldap://server.example.com
backup create
```

- By manually creating the task:

```
# ldapadd -D uid=example,ou=People,dc=example,dc=com -W -H
ldap://server.example.com

dn: cn=backup-2021_07_23_12:55_00,cn=backup,cn=tasks,cn=config
changetype: add
objectClass: extensibleObject
nsarchivedir: /var/lib/dirsrv/slapd-instance_name/bak/backup-2021_07_23_12:55_00
nsdatabasetype: ldbm database
cn: backup-2021_07_23_12:55_00
```

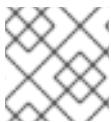
#### Verification

- Verify that the backup was created:

```
# ls -l /var/lib/dirsrv/slapd-instance_name/bak/
total 0
drwx----- 3 dirsrv dirsrv 108 Jul 23 12:55 backup-2021_07_23_12_55_00
...
```

## 6.4. RESTORING DIRECTORY SERVER

In certain situations, administrators want to restore Directory Server, for example, after a hardware failure. This section describes the supported restore methods.

**NOTE**

Directory Server does not support restoring individual databases.

Directory Server runs the restore operation as the **dirsrv** user. Therefore, the permissions of the directory containing the backup must allow this user to read the files.

### 6.4.1. Restoring All Databases Using the Command Line

Directory Server supports restoring databases while the instance is running or while the instance is offline:

- Use one of the following methods if the instance is running:
  - Use the **dsconf backup restore** command. See [Section 6.4.1.1.1, "Restoring All Databases Using the dsconf backup restore Command"](#).
  - Create a **cn=tasks** entry. See [Section 6.4.1.1.2, "Restoring all Databases Using a cn=tasks entry"](#).
- If the instance is offline, use the **dsctl bak2db** command. See [Section 6.4.1.2, "Restoring all Databases While the Server is Offline"](#).

#### 6.4.1.1. Restoring All Databases While the Server is Running

##### 6.4.1.1.1. Restoring All Databases Using the **dsconf backup restore** Command

Use the **dsconf backup restore** command to automatically create a task that restores up all databases from a backup directory.

For example, to restore the backup stored in the **/var/lib/dirsrv/slapd-instance\_name/bak/instance\_name-time\_stamp/** directory:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backup restore /var/lib/dirsrv/slapd-
instance_name/bak/instance_name-time_stamp/
The backup restore task has finished successfully
```

##### 6.4.1.1.2. Restoring all Databases Using **acn=tasks** entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate a restore operation, create a task in the **cn=restore,cn=tasks,cn=config** entry.

**WARNING**

Using a restore task overrides all data in the instance.

A restore task entry requires the following attributes:

- ***cn***: Sets the unique name of the task.
- ***nsArchiveDir***: Sets the path to the directory that contains the backup.
- ***nsDatabaseType***: Sets the type of the database to restore. Directory Server supports only the **ldbm database** value in this attribute.

For example, to add a task that restores all databases from the backup stored in the **/var/lib/dirsrv/slappd-*instance\_name*/bak/*instance\_name-time\_stamp*** directory:

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com -x

dn: cn=example_restore,cn=import,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example_restore
nsArchiveDir: /var/lib/dirsrv/slappd-instance_name/bak/instance_name-time_stamp
nsDatabaseType: ldbm database
```

When the task is completed, the entry is removed from the directory configuration.

#### 6.4.1.2. Restoring all Databases While the Server is Offline

If the server is offline when you restore databases, use the **dsctl bak2db** command:

1. Stop the instance:

```
# dsctl instance_name stop
```

2. Restore the databases. For example, to add a task that restores all databases from the backup stored in the **/var/lib/dirsrv/slappd-*instance\_name*/bak/*instance\_name-time\_stamp*** directory:

```
# dsctl instance_name bak2db
/var/lib/dirsrv/slappd-instance_name/bak/instance_name-time_stamp
bak2db successful
```



#### NOTE

The **dsctl bak2db** command runs as the restore as the **dirsrv** user. Therefore, the permissions of the source directory must allow this user to read files and directories.

3. Start the instance:

```
# dsctl instance_name start
```

#### 6.4.2. Restoring All Databases Using the Web Console

To restore all database using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).

2. Select the instance.
3. Click the **Actions** button, and select **Manage Backups**.

The displayed window lists the available backups in the `/var/lib/dirsrv/slapd-instance_name/bak/` directory.

The screenshot shows a 'Manage Backups' interface. At the top, there's a title bar with 'Manage Backups' and a close button. Below it is a table with three columns: 'Backup ^', 'Creation Date', and 'Size'. A single row is present, showing 'example\_backup', '2020-03-25 10:05:31', and '11M'. To the right of this row is a blue 'Actions ▾' button. Below the table is a pagination section with '6 ^ per page', '1-1 of 1', and navigation buttons. At the bottom are two buttons: 'Create Backup' and 'Refresh Backups'.

Backup ^	Creation Date	Size
example_backup	2020-03-25 10:05:31	11M

6 ^ per page      1-1 of 1      << < 1 of 1 > >>

Create Backup      Refresh Backups

4. Open the **Actions** menu next to the backup you want to restore, and select **Restore Backup**.
5. Click **Yes** to confirm.

#### 6.4.3. Restoring Databases That Include Replicated Entries

Several situations can occur when a supplier server is restored:

- The consumer servers are also restored.

For the very unlikely situation, that all databases are restored from backups taken at exactly the same time (so that the data are in sync), the consumers remain synchronized with the supplier, and it is not necessary to do anything else. Replication resumes without interruption.

- Only the supplier is restored.

If only the supplier is restored or if the consumers are restored from backups taken at a different times, reinitialize the consumers for the supplier to update the data in the database. If only the supplier is restored or if the consumers are restored from backups taken at a different times, reinitialize the consumers for the supplier to update the data in the database.

- Changelog entries have not yet expired on the supplier server.

If the supplier's changelog has not expired since the database backup was taken, then restore the local consumer and continue with normal operations. This situation occurs only if the backup was taken within a period of time that is shorter than the value set for the maximum changelog age attribute, ***nsslapd-changelogmaxage***, in the **cn=changelog5,cn=config** entry. For more information about this option, see the *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, reinitialize the consumer. For more information on reinitializing consumers, see [Section 15.8.3, “Initializing a Consumer”](#).

### Example 6.3. Restoring a Directory Server Replication Topology

For example, to restore all servers in a replication environment, consisting of two suppliers and two consumer server:

1. Restore the first supplier. Use the **dsconf backend import** command to import the data. See [Section 6.1.2, “Importing Using the Command Line”](#).
2. Online-initialize the remaining servers by using replication:
  - a. Initialize the second supplier from the first one.
  - b. Initialize the consumers from the supplier.

For details, see [Section 15.8.3, “Initializing a Consumer”](#).

3. On each server, display the replication status to verify that replication works correctly. For details, see [Section 15.22, “Displaying the Status of a Specific Replication Agreement”](#).

The changelog associated with the restored database will be erased during the restore operation. A message will be logged to the supplier servers' log files indicating that reinitialization is required.

For information on managing replication, see [Chapter 15, Managing Replication](#).

# CHAPTER 7. MANAGING ATTRIBUTES AND VALUES

Red Hat Directory Server provides several different mechanisms for dynamically and automatically maintaining some types of attributes on directory entries. These plug-ins and configuration options simplify managing directory data and expressing relationships between entries.

Part of the characteristic of entries are their *relationships* to each other. Obviously, a manager has an employee, so those two entries are related. Groups are associated with their members. There are less apparent relationships, too, like between entries which share a common physical location.

Red Hat Directory Server provides several different ways that these relationships between entries can be maintained smoothly and consistently. There are several plug-ins can apply or generate attributes automatically as part of the data within the directory, including classes of service, linking attributes, and generating unique numeric attribute values.

## 7.1. ENFORCING ATTRIBUTE UNIQUENESS

To ensure that the value of an attribute is unique across the directory or subtree, use the **Attribute Uniqueness** plug-in.

If you want multiple attributes to be unique or if you want to use different conditions, create multiple configuration records of the plug-in.

### 7.1.1. Creating a New Configuration Record of the Attribute Uniqueness Plug-in

For each attribute whose values must be unique, create a new configuration record of the **Attribute Uniqueness** plug-in.



#### NOTE

You can only create a new configuration record of the plug-in from the command line.

To create a new unconfigured and disabled configuration record of the plug-in named **Example Attribute Uniqueness**:

```
dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq add "Example" --attr-name uid
```

### 7.1.2. Configuring Attribute Uniqueness over Suffixes or Subtrees

You can configure the **Attribute Uniqueness** plug-in to ensure that values of an attribute are unique in certain suffixes, subtrees, or over suffixes and subtrees.

#### 7.1.2.1. Configuring Attribute Uniqueness over Suffixes or Subtrees Using the Command Line

To configure, for example, that values stored in **mail** attributes are unique:

1. Create a new configuration record of the **Attribute Uniqueness** plug-in named, for example, **mail Attribute Uniqueness**. For details, see [Section 7.1.1, “Creating a New Configuration Record of the Attribute Uniqueness Plug-in”](#).
2. Enable the plug-in configuration record:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq enable "mail Attribute Uniqueness"
```

3. Configure that values stored in **mail** attributes must be unique inside, for example, the **ou=Engineering,dc=example,dc=com** and **ou=Sales,dc=example,dc=com** subtrees:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq set "mail Attribute Uniqueness" --attr-name mail --subtree ou=Engineering,dc=example,dc=com ou=Sales,dc=example,dc=com
```

4. Optionally, to configure uniqueness across all subtrees configured in this plug-in configuration record:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq set "mail Attribute Uniqueness" --across--all-subtrees=on
```

5. Restart the instance:

```
# dsctl instance_name restart
```

### 7.1.2.2. Configuring Attribute Uniqueness over Suffixes or Subtrees Using the Web Console

To configure, for example, that values stored in **mail** attributes are unique:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Attribute Uniqueness** plug-in.
5. Click **Add Config**.
6. Fill the fields, and enable the config. For example:

**Add Attribute Uniqueness Plugin Config Entry**

Config Name	mail Attribute Uniqueness
Attribute Names	mail
Subtrees	ou=Engineering,dc=example,dc=com ou=Sales,dc=example,dc=com
Top Entry OC	Type an objectClass...
Subtree Entries OC	Type an objectClass... <input checked="" type="checkbox"/> Across All Subtrees
Enable config	ON <input type="checkbox"/>
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

Figure 7.1. Adding an Attribute Uniqueness Configuration

7. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 7.1.3. Configuring Attribute Uniqueness over Object Classes

You can configure the **Attribute Uniqueness** plug-in to ensure that values of an attribute are unique in subtree entries that contain a specific object class. Directory Server searches for this object class in the parent entry of the updated object. If Directory Server did not find the object class, the search continues at the next higher level entry up to the root of the directory tree. If the object class was found, Directory Server verifies that the value of the attribute set in **uniqueness-attribute-name** is unique in this subtree.

To configure, for example, that values stored in **mail** attributes are unique under the entry that contains the **nsContainer** object class:

1. Create a new configuration record of the **Attribute Uniqueness** plug-in named, for example, **mail Attribute Uniqueness**. For details, see [Section 7.1.1, “Creating a New Configuration Record of the Attribute Uniqueness Plug-in”](#).
2. Enable the plug-in configuration record:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq enable "mail Attribute Uniqueness"
```

3. Configure that values stored in **mail** attributes must be unique under the entry that contains the **nsContainer** object class:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq set "mail Attribute Uniqueness" --top-entry-oc=nsContainer
```

4. Optionally, you can limit the scope of objects being checked. If you want the server to check only a subset of entries under the entry that contains the **nsContainer** object class, set an additional object class in the **uniqueness-subtree-entries-oc** parameter. This additional class will also have to be present.

For example, the **mail** attribute must be unique in all entries under the entry that contains the **nsContainer** object class set. However, you want that the plug-in only searches the **mail** in entries that contain a object class that provides this attribute, such as **inetOrgPerson**. In this situation enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin attr-uniq set "mail Attribute Uniqueness" --subtree-entries-oc/inetOrgPerson
```

5. Restart the instance:

```
# dsctl instance_name restart
```

#### 7.1.4. Attribute Uniqueness Plug-in Configuration Parameters

To configure an **Attribute Uniqueness** plug-in configuration record, set the plug-in's configuration attributes in the **cn=attribute\_uniqueness\_configuration\_record\_name,cn=plugins,cn=config** entry.

##### Example 7.1. Attribute Uniqueness Plug-in Configuration Using Plug-in-specific Attributes

```
dn: cn=Example Attribute Uniqueness,cn=plugins,cn=config
nsslapd-pluginEnabled: on
uniqueness-attribute-name: attribute_name
uniqueness-top-entry-oc: objectclass1
uniqueness-subtree-entries-oc: objectclass2
```

For a list of parameters you can set to configure the **Attribute Uniqueness** plug-in, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

## 7.2. ASSIGNING CLASS OF SERVICE

A *class of service definition* (CoS) shares attributes between entries in a way that is transparent to applications. CoS simplifies entry management and reduces storage requirements.

Clients of the Directory Server read the attributes in a user's entry. With CoS, some attribute values may not be stored within the entry itself. Instead, these attribute values are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of two types of entry in the directory:

- *CoS definition entry*. The CoS definition entry identifies the type of CoS used. Like the role definition entry, it inherits from the **LDAPsubentry** object class. The CoS definition entry is below the branch at which it is effective.

- *Template entry.* The CoS template entry contains a list of the shared attribute values. Changes to the template entry attribute values are automatically applied to all the entries within the scope of the CoS. A single CoS might have more than one template entry associated with it.

The CoS definition entry and template entry interact to provide attribute information to their target entries, any entry within the scope of the CoS.

### 7.2.1. About the CoS Definition Entry

The CoS definition entry is an instance of the **cosSuperDefinition** object class. The CoS definition entry also contains one of three object class that specifies the type of template entry it uses to generate the entry. The target entries which interact with the CoS share the same parent as the CoS definition entry.

There are three types of CoS, defined using three types of CoS definition entries:

- *Pointer CoS.* A pointer CoS identifies the template entry using the template DN only.
- *Indirect CoS.* An indirect CoS identifies the template entry using the value of one of the target entry's attributes. For example, an indirect CoS might specify the **manager** attribute of a target entry. The value of the **manager** attribute is then used to identify the template entry.

The target entry's attribute must be single-valued and contain a DN.

- *Classic CoS.* A classic CoS identifies the template entry using a combination of the template entry's base DN and the value of one of the target entry's attributes.

For more information about the object classes and attributes associated with each type of CoS, see [Section 7.2.10, “Managing CoS from the Command Line”](#).

If the CoS logic detects that an entry contains an attribute for which the CoS is generating values, the CoS, by default, supplies the client application with the attribute value in the entry itself. However, the CoS definition entry can control this behavior.

### 7.2.2. About the CoS Template Entry

The CoS template entry contains the value or values of the attributes generated by the CoS logic. The CoS template entry contains a general object class of **cosTemplate**. The CoS template entries for a given CoS are stored in the directory tree along with the CoS definition.

The relative distinguished name (RDN) of the template entry is determined by one of the following:

- The DN of the template entry alone. This type of template is associated with a pointer CoS definition.
- The value of one of the target entry's attributes. The attribute used to provide the relative DN to the template entry is specified in the CoS definition entry using the **cosIndirectSpecifier** attribute. This type of template is associated with an indirect CoS definition.
- By a combination of the DN of the subtree where the CoS performs a one level search for templates and the value of one of the target entry's attributes. This type of template is associated with a classic CoS definition.

### 7.2.3. How a Pointer CoS Works

An administrator creates a pointer CoS that shares a common postal code with all of the entries stored under **dc=example,dc=com**. The three entries for this CoS appear as illustrated in [Figure 7.2, "Sample Pointer CoS"](#).

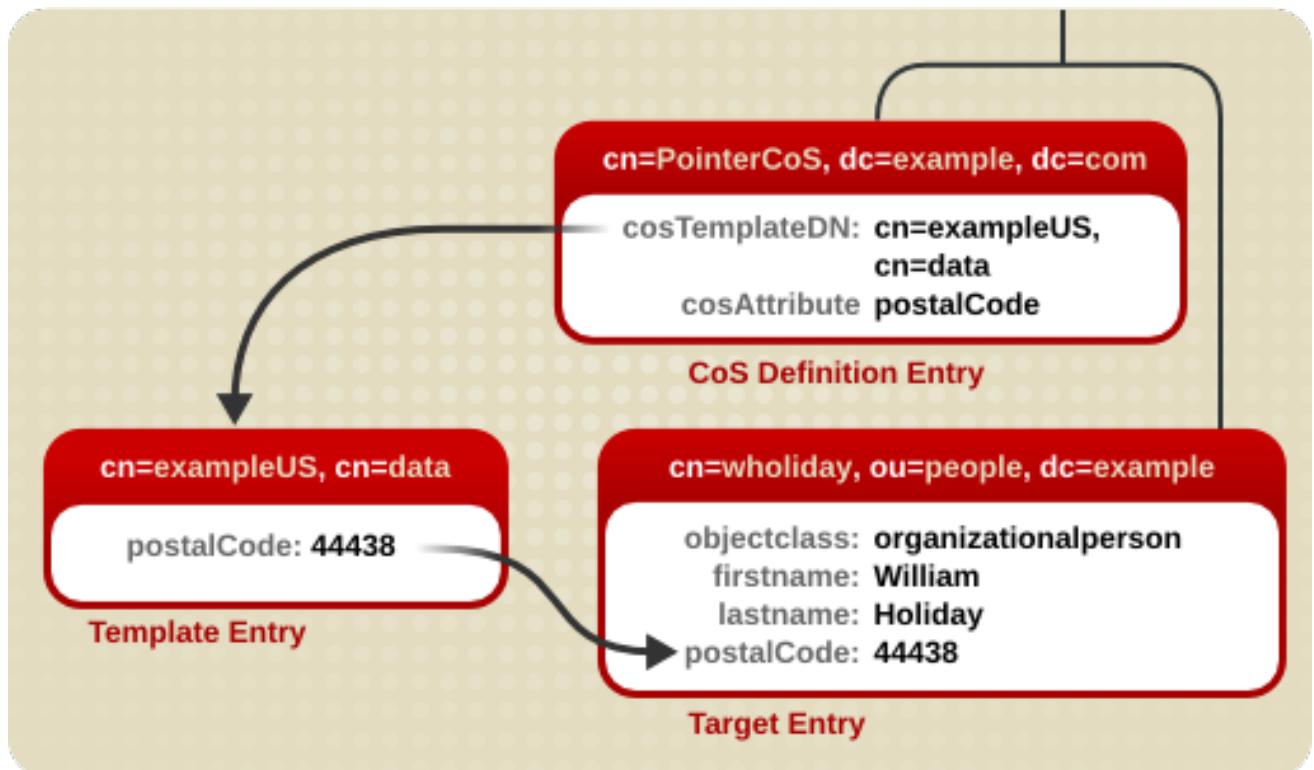


Figure 7.2. Sample Pointer CoS

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**.

#### 7.2.4. How an Indirect CoS Works

An administrator creates an indirect CoS that uses the **manager** attribute of the target entry to identify the template entry. The three CoS entries appear as illustrated in [Figure 7.3, "Sample Indirect CoS"](#).

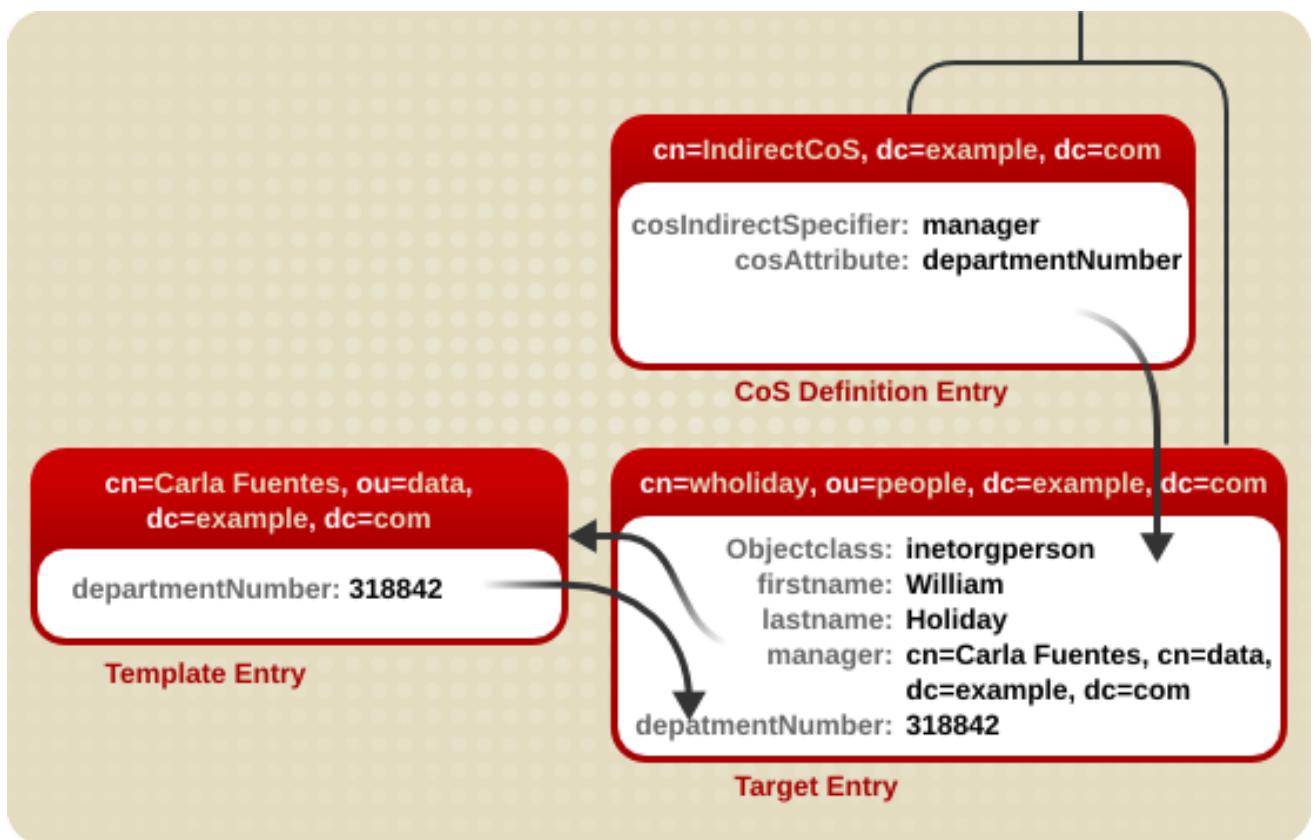


Figure 7.3. Sample Indirect CoS

In this example, the target entry for William Holiday contains the indirect specifier, the **manager** attribute. William's manager is Carla Fuentes, so the **manager** attribute contains a pointer to the DN of the template entry, **cn=Carla Fuentes,ou=people,dc=example,dc=com**. The template entry in turn provides the **departmentNumber** attribute value of **318842**.

### 7.2.5. How a Classic CoS Works

An administrator creates a classic CoS that uses a combination of the template DN and a CoS specifier to identify the template entry containing the postal code. The three CoS entries appear as illustrated in Figure 7.4, "Sample Classic CoS":

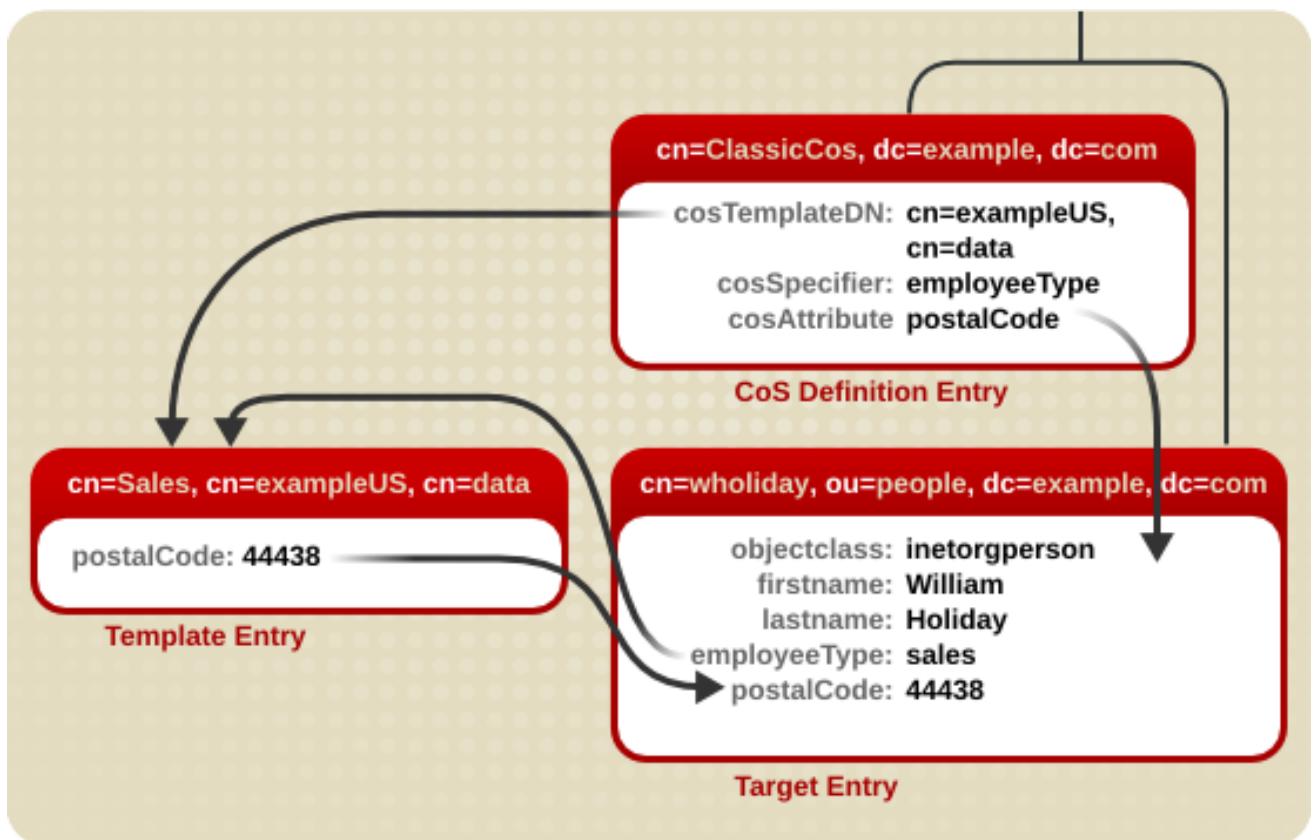


Figure 7.4. Sample Classic CoS

In this example, the CoS definition entry's **cosSpecifier** attribute specifies the **employeeType** attribute. This attribute, in combination with the template DN, identify the template entry as **cn=sales,cn=exampleUS,cn=data**. The template entry then provides the value of the **postalCode** attribute to the target entry.

### 7.2.6. Handling Physical Attribute Values

The **cosAttribute** attribute contains the name of another attribute which is governed by the class of service. This attribute allows an *override* qualifier after the attribute value which sets how the CoS handles existing attribute values on entries when it generates attribute values.

**cosAttribute: attribute\_name override**

There are four *override* qualifiers:

- **default:** Only returns a generated value if there is no corresponding attribute value stored with the entry.
- **override:** Always returns the value generated by the CoS, even when there is a value stored with the entry.
- **operational:** Returns a generated attribute only if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order to be returned. When **operational** is used, it also overrides any existing attribute values.

**NOTE**

An attribute can only be made operational if it is defined as operational in the schema. For example, if the CoS generates a value for the ***description*** attribute, it is not possible to use the ***operational*** qualifier because this attribute is not marked operational in the schema.

- ***operational-default***: Only returns a generated value if there is no corresponding attribute value stored with the entry and if it is explicitly requested in the search.

If no qualifier is set, ***default*** is assumed.

For example, this pointer CoS definition entry indicates that it is associated with a template entry, ***cn=exampleUS,ou=data,dc=example,dc=com***, that generates the value of the ***postalCode*** attribute. The ***override*** qualifier indicates that this value will take precedence over the value stored by the entries for the ***postalCode*** attribute:

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode override
```

**NOTE**

If an entry contains an attribute value generated by a CoS, the value of the attribute *cannot* be manually updated if it is defined with the ***operational*** or ***override*** qualifiers.

For more information about the CoS attributes, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

### 7.2.7. Handling Multi-valued Attributes with CoS

Any attribute can be generated using a class of service – including multi-valued attributes. That introduces the potential for confusion. Which CoS supplies a value? Any of them or all of them? How is the value selected from competing CoS templates? Does the generated attribute use a single value or multiple values?

There are two ways to resolve this:

- Creating a rule to merge multiple CoS-generated attributes into the target entry. This results in multiple values in the target entry.
- Setting a priority to select one CoS value out of competing CoS definitions. This generates one single value for the target entry.

**NOTE**

Indirect CoS do not support the ***cosPriority*** attribute.

The way that the CoS handles multiple values for a CoS attribute is defined in whether it uses a *merge-schemes* qualifier.

**cosAttribute: attribute override merge-schemes**



### NOTE

The *merge-schemes* qualifier does not affect how the CoS handles physical attribute values or the *override* qualifier. If there are multiple competing CoS templates or definitions, then the same *merge-schemes* and *override* qualifiers have to be set on every **cosAttribute** for every competing CoS definition. Otherwise, one combination is chosen arbitrarily from all possible CoS definitions.

Using the *merge-schemes* qualifier tells the CoS that it will, or can, generate multiple values for the managed attribute. There are two possible scenarios for having a multi-valued CoS attribute:

- One CoS template entry contains multiple instances of the managed CoS attribute, resulting in multiple values on the target entry. For example:

```
dn: cn=server access template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: mail.example.com
accessTo: irc.example.com
```



### NOTE

This method only works with classic CoS.

- Multiple CoS definitions may define a class of service for the same target attribute, so there are multiple template entries. For example:

```
dn: cn=mail template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: mail.example.com

dn: cn=chat template,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
accessTo: irc.example.com
```

However, it may be that even if there are multiple CoS definitions, only one value should be generated for the attribute. If there are multiple CoS definitions, then the value is chosen arbitrarily. This is an unpredictable and unwieldy option. The way to control which CoS template to use is to set a ranking on the template – a *priority* – and the highest prioritized CoS always "wins" and provides the value.

It is fairly common for there to be multiple templates completing to provide a value. For example, there can be a multi-valued **cosSpecifier** attribute in the CoS definition entry. The template priority is set using the **cosPriority** attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

For example, a CoS template entry for generating a department number appears as follows:

```
dn: cn=data,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
departmentNumber: 71776
cosPriority: 0
```

This template entry contains the value for the **departmentNumber** attribute. It has a priority of zero, meaning this template takes precedence over any other conflicting templates that define a different **departmentNumber** value.

Templates that contain no **cosPriority** attribute are considered the lowest priority. Where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.



#### NOTE

The behavior for negative **cosPriority** values is not defined in Directory Server; do not enter negative values.

### 7.2.8. Searches for CoS-Specified Attributes

CoS definitions provide values for attributes in entries. For example, a CoS can set the **postalCode** attribute for every entry in a subtree. Searches against those CoS-defined attributes, however, do not behave like searches against regular entries.

If the CoS-defined attribute is indexed with any kind of index (including presence), then any attribute with a value set by the CoS is not returned with a search. For example:

- The **postalCode** attribute for Ted Morris is defined by a CoS.
- The **postalCode** attribute for Barbara Jensen is set in her entry.
- The **postalCode** attribute is indexed.

If an **ldapsearch** command uses the filter **(postalCode=\*)**, then Barbara Jensen's entry is returned, while Ted Morris's is not.

If the CoS-defined attribute is *not* indexed, then every matching entry is returned in a search, regardless of whether the attribute value is set locally or with CoS. For example:

- The **postalCode** attribute for Ted Morris is defined by a CoS.
- The **postalCode** attribute for Barbara Jensen is set in her entry.
- The **postalCode** attribute is *not* indexed.

If an **ldapsearch** command uses the filter **(postalCode=\*)**, then both Barbara Jensen's and Ted Morris's entries are returned.

CoS allows for an *override*, an identifier given to the **cosAttribute** attribute in the CoS entry, which means that local values for an attribute can override the CoS value. If an override is set on the CoS, then an **ldapsearch** operation will return a value for an entry even if the attribute is indexed, as long as there

is a local value for the entry. Other entries which possess the CoS but do not have a local value will still not be returned in the **ldapsearch** operation.

Because of the potential issues with running LDAP search requests on CoS-defined attributes, take care when deciding which attributes to generate using a CoS.

## 7.2.9. Access Control and CoS

The server controls access to attributes generated by a CoS in exactly the same way as regular stored attributes. However, access control rules depending upon the value of attributes generated by CoS will not work. This is the same restriction that applies to using CoS-generated attributes in search filters.

## 7.2.10. Managing CoS from the Command Line

Because all configuration information and template data is stored as entries in the directory, standard LDAP tools can be used for CoS configuration and management.

- [Section 7.2.10.1, "Creating the CoS Definition Entry from the Command Line"](#)
- [Section 7.2.10.2, "Creating the CoS Template Entry from the Command Line"](#)
- [Section 7.2.10.3, "Example of a Pointer CoS"](#)
- [Section 7.2.10.4, "Example of an Indirect CoS"](#)
- [Section 7.2.10.5, "Example of a Classic CoS"](#)
- [Section 7.2.10.6, "Searching for CoS Entries"](#)

### 7.2.10.1. Creating the CoS Definition Entry from the Command Line

Each type of CoS requires a particular object class to be specified in the definition entry. All CoS definition object classes inherit from the **LDAPsubentry** object class and the **cosSuperDefinition** object class.

A pointer CoS uses the **cosPointerDefinition** object class. This object class identifies the template entry using an entry DN value specified in the **cosTemplateDn** attribute, as shown in [Example 7.2, "An Example Pointer CoS Entry"](#).

#### Example 7.2. An Example Pointer CoS Entry

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn:DN_string
cosAttribute:list_of_attributes qualifier
cn: pointerCoS
```

An indirect CoS uses the **cosIndirectDefinition** object class. This type of CoS identifies the template entry based on the value of one of the target entry's attributes, as specified in the **cosIndirectSpecifier** attribute. This is illustrated in [Example 7.3, "An Example Indirect CoS Entry"](#).

**Example 7.3. An Example Indirect CoS Entry**

```
dn: cn=indirectCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier:attribute_name
cosAttribute:list_of_attributes qualifier
cn: indirectCoS
```

A classic CoS uses the **cosClassicDefinition** object class. This identifies the template entry using both the template entry's DN (set in the **cosTemplateDn** attribute) and the value of one of the target entry's attributes (set in the **cosSpecifier** attribute). This is illustrated in [Example 7.4, "An Example Classic CoS Entry"](#).

**Example 7.4. An Example Classic CoS Entry**

```
dn: cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn:DN_string
cosSpecifier:attribute_name
cosAttribute:list_of_attributes qualifier
cn: classicCoS
```

For a class of service, the object class defines the type of CoS, and the supporting attributes identify which directory entries are affected by defining the CoS template. Every CoS has one additional attribute which can be defined for it: **cosAttribute**. The purpose of a CoS is to supply attribute values across multiple entries; the **cosAttribute** attribute defines which attribute the CoS generates values for.

**7.2.10.2. Creating the CoS Template Entry from the Command Line**

Each template entry is an instance of the **cosTemplate** object class.

**NOTE**

Consider adding the **LDAPsubentry** object class to a new template entry. Making the CoS template entry an instance of the **LDAPsubentry** object classes allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else, such as a user entry, the **LDAPsubentry** object class does not need to be added to the template entry.

The CoS template entry also contains the attribute generated by the CoS (as specified in the **cosAttribute** attribute of the CoS definition entry) and the value for that attribute.

For example, a CoS template entry that provides a value for the **postalCode** attribute follows:

```
dn:cn=exampleUS,ou=data,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
```

```
objectclass: cosTemplate
postalCode: 44438
```

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

- [Section 7.2.10.3, "Example of a Pointer CoS"](#)
- [Section 7.2.10.4, "Example of an Indirect CoS"](#)
- [Section 7.2.10.5, "Example of a Classic CoS"](#)

### 7.2.10.3. Example of a Pointer CoS

Example Corporation's administrator is creating a pointer CoS that shares a common postal code with all entries in the **dc=example,dc=com** tree.

1. Add a new pointer CoS definition entry to the **dc=example,dc=com** suffix using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=pointerCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode
```

2. Create the template entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=exampleUS,ou=data,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438
```

The CoS template entry (**cn=exampleUS,ou=data,dc=example,dc=com**) supplies the value stored in its **postalCode** attribute to any entries located under the **dc=example,dc=com** suffix. These entries are the target entries.

### 7.2.10.4. Example of an Indirect CoS

This indirect CoS uses the **manager** attribute of the target entry to identify the CoS template entry, which varies depending on the different values of the attribute.

1. Add a new indirect CoS definition entry to the **dc=example,dc=com** suffix using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=indirectCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
```

```
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

If the directory or modify the manager entries already contain the **departmentNumber** attribute, then no other attribute needs to be added to the manager entries. The definition entry looks in the target suffix (the entries under **dc=example,dc=com**) for entries containing the **manager** attribute because this attribute is specified in the **cosIndirectSpecifier** attribute of the definition entry). It then checks the **departmentNumber** value in the manager entry that is listed. The value of the **departmentNumber** attribute will automatically be relayed to all of the manager's subordinates that have the **manager** attribute. The value of **departmentNumber** will vary depending on the department number listed in the different manager's entries.

#### 7.2.10.5. Example of a Classic CoS

The Example Corporation administrator is creating a classic CoS that automatically generates postal codes using a combination of the template DN and the attribute specified in the **cosSpecifier** attribute.

1. Add a new classic CoS definition entry to the **dc=example,dc=com** suffix using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=classicCoS,dc=example,dc=com
cosSpecifier: businessCategory
cosAttribute: postalCode override
```

2. Create the template entries for the sales and marketing departments. Add the CoS attributes to the template entry. The **cn** of the template sets the value of the **businessCategory** attribute in the target entry, and then the attributes are added or overwritten according to the value in the template:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=sales,cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438
-
dn: cn=marketing,cn=classicCoS,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 99111
```

The classic CoS definition entry applies to all entries under the **dc=example,dc=com** suffix. Depending upon the combination of the **businessCategory** attribute found in the entry and the **cosTemplateDn**, it can arrive at one of two templates. One, the sales template, provides a postal code specific to employees in the sales department. The marketing template provides a postal code specific to employees in the marketing department.

### 7.2.10.6. Searching for CoS Entries

CoS definition entries are *operational* entries and, by default, not returned in regular searches. To return CoS definition entries in searches, add the **IdapSubEntry** object class to the CoS definition entries. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=pointerCoS,ou=People,dc=example,dc=com
changetype: add
objectclass: ldapSubEntry
```

Then use the (**objectclass=ldapSubEntry**) filter with the **Idapssearch** utility to search for entries containing the **IdapSubEntry** object class. For example:

```
# Idapssearch -x -s sub -b ou=People,dc=example,dc=com "(|(objectclass=*)(objectclass=ldapSubEntry))"
```

This search returns all regular entries in addition to CoS definition entries in the **ou=People,dc=example,dc=com** subtree.

### 7.2.11. Creating Role-Based Attributes

Classic CoS schemes generate attribute values for an entry based on the role possessed by the entry. For example, role-based attributes can be used to set the server look-through limit on an entry-by-entry basis.

To create a role-based attribute, use the **nsRole** attribute as the **cosSpecifier** in the CoS definition entry of a classic CoS. Because the **nsRole** attribute can be multi-valued, CoS schemes can be defined that have more than one possible template entry. To resolve the ambiguity of which template entry to use, include the **cosPriority** attribute in the CoS template entry.

For example, this CoS allows members of the manager role to exceed the standard mailbox quota. The manager role entry is:

```
dn: cn=ManagerRole,ou=people,dc=example,dc=com
objectclass: top
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: ManagerRole
nsRoleFilter: ou=managers
Description: filtered role for managers
```



#### IMPORTANT

The **nsRoleFilter** attribute cannot accept virtual attribute values.

The classic CoS definition entry looks like:

```
dn: cn=managerCOS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
```

```
cosTemplateDn: cn=managerCOS,dc=example,dc=com
cosSpecifier: nsRole
cosAttribute: mailboxquota override
```

The ***cosTemplateDn*** attribute provides a value that, in combination with the attribute specified in the ***cosSpecifier*** attribute (in the example, the ***nsRole*** attribute of the target entry), identifies the CoS template entry. The CoS template entry provides the value for the ***mailboxquota*** attribute. An additional qualifier of ***override*** tells the CoS to override any existing ***mailboxquota*** attributes values in the target entry.

The corresponding CoS template entry looks as follows:

```
dn:cn="cn=ManagerRole,ou=people,dc=example,dc=com",cn=managerCOS,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
mailboxquota: 1000000
```

The template provides the value for the ***mailboxquota*** attribute, **1000000**.



### NOTE

The role entry and the CoS definition and template entries should be located at the same level in the directory tree.

## 7.3. LINKING ATTRIBUTES TO MANAGE ATTRIBUTE VALUES

A class of service dynamically supplies attribute values for entries which all have attributes with the *same value*, like building addresses, postal codes, or main office numbers. These are shared attribute values, which are updated in a single template entry.

Frequently, though, there are relationships between entries where there needs to be a way to express linkage between them, but the values (and possibly even the attributes) that express that relationship are different. Red Hat Directory Server provides a way to link specified attributes together, so that when one attribute in one entry is altered, a corresponding attribute on a related entry is automatically updated. (The link and managed attributes both have DN values. The value of the link attribute contains the DN of the entry for the plug-in to update; the managed attribute in the second entry has a DN value which points back to the original link entry.)

### 7.3.1. About Linking Attributes

The Linked Attributes Plug-in, allows multiple instances of the plug-in. Each instance configures one attribute which is manually maintained by the administrator (***linkType***) and one attribute which is automatically maintained by the plug-in (***managedType***).

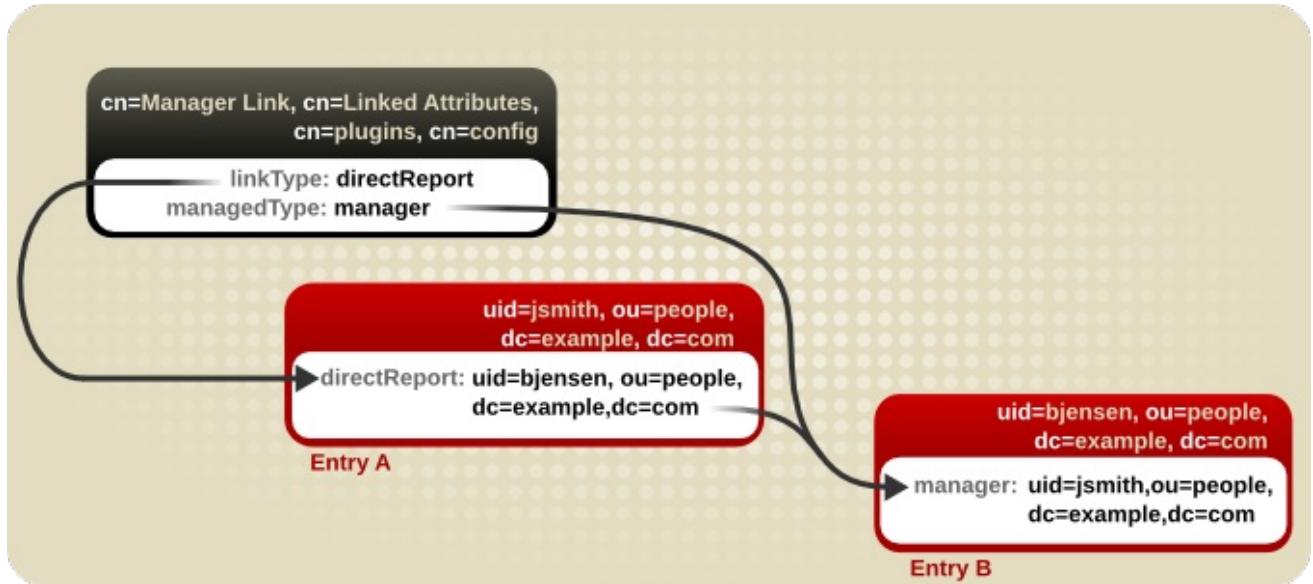


Figure 7.5. Basic Linked Attribute Configuration



#### NOTE

To preserve data consistency, only the plug-in process should maintain the managed attribute. Consider creating an ACI that will restrict all write access to any managed attribute. See [Section 18.7.2, "Adding an ACI"](#) for information on setting ACIs.

A Linked Attribute Plug-in instance can be restricted to a single subtree within the directory. This can allow more flexible customization of attribute combinations and affected entries. If no scope is set, then the plug-in operates in the entire directory.

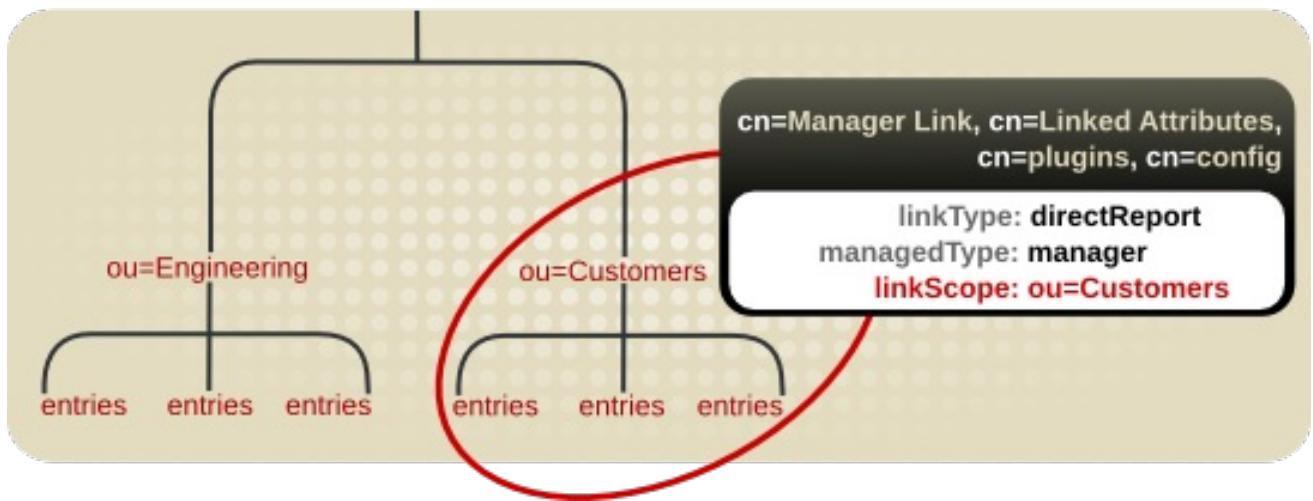


Figure 7.6. Restricting the Linked Attribute Plug-in to a Specific Subtree

When configuring the Linked Attribute Plug-in instance, certain configurations are required:

- Both the managed attribute and linked attribute must require the Distinguished Name syntax in their attribute definitions. The linked attributes are essentially managed cross-references, and the way that the plug-in handles these cross-references is by pulling the DN of the entry from the attribute value.

For information on planning custom schema elements, see [Chapter 12, Managing the Directory Schema](#).

- Each Linked Attribute Plug-in instance must be local and any *managed* attributes must be blocked from replication using fractional replication.

Any changes that are made on one supplier will automatically trigger the plug-in to manage the values on the corresponding directory entries, so the data stay consistent across servers.

However, the managed attributes must be maintained by the plug-in instance for the data to be consistent between the linked entries. This means that managed attribute values should be maintained solely by the plug-in processes, not the replication process, even in a multi-supplier replication environment.

For information on using fractional replication, see [Section 15.1.7, “Replicating a Subset of Attributes with Fractional Replication”](#).

### 7.3.2. Looking at the Linking Attributes Plug-in Syntax

The default Linked Attributes Plug-in entry is a container entry for each plug-in instance, similar to the password syntax plug-ins or the DNA Plug-in in the next section. Each entry beneath this container entry defines a different link-managed attribute pair.

To create a new linking attribute pair, then, create a new plug-in instance beneath the container entry. A basic linking attribute plug-in instance required defining two things:

- The attribute that is managed manually by administrators, in the ***linkType*** attribute
- The attribute that is created dynamically by the plug-in, in the ***managedType*** attribute
- Optionally, a scope that restricts the plug-in to a specific part of the directory tree, in the ***linkScope*** attribute

#### Example 7.5. Example Linked Attributes Plug-in Instance Entry

```
dn: cn=Manager Link,cn=Linked Attributes,cn=plugins,cn=config
objectClass: top
objectClass: extensibleObject
cn: Manager Link
linkType: directReport
managedType: manager
linkScope: ou=people,dc=example,dc=com
```

For a list of attributes available for an instance of the **Linked Attributes** plug-in, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

### 7.3.3. Configuring Attribute Links

1. If it is not already enabled, enable the **Linked Attributes** plug-in. For details, see [Section 1.10.3, “Enabling and Disabling Plug-ins”](#).
2. Create the plug-in instance. Both the **--managedType** and **--link-type** parameters are required. The following example shows the plug-in instance created by using **dsconf**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin linked-attr config
"Manager Link" add --link-type=directReport --managed-type=manager
```

3. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 7.3.4. Cleaning up Attribute Links

The managed-linked attributes can get out of sync. For instance, a linked attribute could be imported or replicated over to a server, but the corresponding managed attribute was not because the link attribute was not properly configured. The managed-linked attribute pairs can be fixed by running the **dsconf plugin linked-attr fixup** command or by launching a fix-up task.

The fixup task removes any managed attributes (attributes managed by the plug-in) that do not have a corresponding link attribute (attributes managed by the administrator) on the referenced entry. Conversely, the task adds any missing managed attributes if the link attribute exists in an entry.

#### 7.3.4.1. Regenerating Linked Attributes

The **dsconf plugin linked-attr fixup** command launches a special task to regenerate all of the managed-link attribute pairs on directory entries. One or the other may be lost in certain situations. If the link attribute exists in an entry, the task traces the cross-referenced DN in the available attribute and creates the corresponding configured managed attribute on the referenced entry. If a managed attribute exists with no corresponding link attribute, then the managed attribute value is removed.

To repair all configured link attribute pairs for the entire scope of the plug-in, then run the command as the Directory Manager:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin linked-attr fixup
```

It is also possible to limit the fixup task to a single link-managed attribute pair by passing a base DN to the command. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin linked-attr fixup "cn=Manager  
Link,cn=Linked Attributes,cn=plugins,cn=config"
```

#### 7.3.4.2. Regenerating Linked Attributes Using Idapmodify

Repairing linked attributes is one of the tasks which can be managed through a special task configuration entry. Task entries occur under the **cn=tasks** configuration entry in the **dse.ldif** file, so it is also possible to initiate a task by adding the entry using **Idapmodify**. When the task is complete, the entry is removed from the directory.

This task is the same one created automatically by the **dsconf plugin linked-attr fixup** command when it is run.

To initiate a linked attributes fixup task, add an entry under the **cn=fixup linked attributes,cn=tasks,cn=config** entry. The only required attribute is the **cn** for the specific task, though it also allows the **ttl** attribute to set a timeout period. Using **Idapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
dn: cn=example,cn=fixup linked attributes,cn=tasks,cn=config  
changetype: add
```

```
cn:example
ttl: 5
```

Once the task is completed, the entry is deleted from the **dse.ldif** configuration, so it is possible to reuse the same task entry continually.

The **cn=fixup linked attributes** task configuration is described in more detail in the [Configuration, Command, and File Reference](#).

## 7.4. ASSIGNING AND MANAGING UNIQUE NUMERIC ATTRIBUTE VALUES

Some entry attributes require having a unique number, such as **uidNumber** and **gidNumber**. The Directory Server can automatically generate and supply unique numbers for specified attributes using the Distributed Numeric Assignment (DNA) Plug-in.



### NOTE

*Attribute uniqueness* is not necessarily preserved with the DNA Plug-in. The plug-in only assigns non-overlapping ranges, but it does allow manually-assigned numbers for its managed attributes, and it does not verify or require that the manually-assigned numbers are unique.

The issue with assigning unique numbers is not with generating the numbers but in effectively avoiding replication conflicts. The DNA Plug-in assigns unique numbers across a *single* back end. For multi-supplier replication, when each supplier is running a local DNA Plug-in instance, there has to be a way to ensure that each instance is using a truly unique set of numbers. This is done by assigning different *ranges* of numbers to each server to assign.

### 7.4.1. About Dynamic Number Assignments

The DNA Plug-in for a server assigns a range of available numbers that that instance can issue. The range definition is very simple and is set by two attributes: the server's next available number (the low end of the range) and its maximum value (the top end of the range). The initial bottom range is set when the plug-in instance is configured. After that, the bottom value is updated by the plug-in. By breaking the available numbers into separate ranges on each replica, the servers can all continually assign numbers without overlapping with each other.

#### 7.4.1.1. Filters, Searches, and Target Entries

The server performs a sorted search, internally, to see if the next specified range is already taken, requiring the managed attribute to have an equality index with the proper ordering matching rule (as described in [Section 13.2, "Creating Standard Indexes"](#)).

The DNA Plug-in is applied, always, to a specific area of the directory tree (the *scope*) and to specific entry types within that subtree (the *filter*).



### IMPORTANT

The DNA Plug-in only works on a single back end; it cannot manage number assignments for multiple databases. The DNA plug-in uses the sort control when checking whether a value has already been manually allocated outside of the DNA Plug-in. This validation, using the sort control, only works on a single back end.

### 7.4.1.2. Ranges and Assigning Numbers

There are several different ways that the Directory Server can handle generating attribute values:

- In the simplest case, a user entry is added to the directory with an object class which requires the unique-number attribute, but without the attribute present. Adding an entry with no value for the managed attribute triggers the DNA Plug-in to assign a value. This option only works if the DNA Plug-in has been configured to assign unique values to a single attribute.
- A similar and more manageable option is to use a *magic number*. This magic number is a template value for the managed attribute, something outside the server's range, a number or even a word, that the plug-in recognizes it needs to replace with a new assigned value. When an entry is added with the magic value and the entry is within the scope and filter of the configured DNA Plug-in, then using the magic number automatically triggers the plug-in to generate a new value. The following example, based on using **ldapmodify**, adds 0 as a magic number:

```
dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
uidNumber: 0
gidNumber: 0
....
```

The DNA Plug-in only generates new, unique values. If an entry is added or modified to use a specific value for an attribute controlled by the DNA Plug-in, the specified number is used; the DNA Plug-in will not overwrite it.

### 7.4.1.3. Multiple Attributes in the Same Range

The DNA Plug-in can assign unique numbers to a single attribute type or across multiple attribute types from a single range of unique numbers.

This provides several options for assigning unique numbers to attributes:

- A single number assigned to a single attribute type from a single range of unique numbers.
- The same unique number assigned to two attributes for a single entry.
- Two different attributes assigned two different numbers from the same range of unique numbers.

In many cases, it is sufficient to have a unique number assigned per attribute type. When assigning an **employeeID** to a new employee entry, it is important each employee entry is assigned a unique **employeeID**.

However, there are cases where it may be useful to assign unique numbers from the same range of numbers to multiple attributes. For example, when assigning a **uidNumber** and a **gidNumber** to a **posixAccount** entry, the DNA Plug-in will assign the same number to both attributes. To do this, then pass both managed attributes to the modify operation, specifying the magic value. Using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -x
```

```

dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
add: uidNumber
uidNumber: 0
-
add:gidNumber
gidNumber: 0

```

When multiple attributes are handled by the DNA Plug-in, the plug-in can assign a unique value to only one of those attributes if the object class only allows one of them. For example, the **posixGroup** object class does not allow a **uidNumber** attribute but it does allow **gidNumber**. If the DNA Plug-in manages both **uidNumber** and **gidNumber**, then when a **posixGroup** entry is created, a unique number for **gidNumber** is assigned from the same range as the **uidNumber** and **gidNumber** attributes. Using the same pool for all attributes managed by the plug-in keeps the assignment of unique numbers aligned and prevents situations where a **uidNumber** and a **gidNumber** on different entries are assigned from different ranges and result in the same *unique* number.

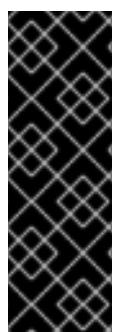
If multiple attributes are handled by the DNA Plug-in, then the same value will be assigned to all of the given managed attributes in an entry in a single modify operation. To assign *different* numbers from the same range, then you must perform separate modify operations. The following example uses **ldapmodify** to do so:

```

# ldapmodify -D "cn=Directory Manager" -W -x
dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
add: uidNumber
uidNumber: 0
^D

# ldapmodify -D "cn=Directory Manager" -W -x
dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
add: employeeld
employeeld: magic

```



## IMPORTANT

When the DNA Plug-in is configured to assign unique numbers to multiple attributes, it is necessary to specify the magic value for each attribute that requires the unique number. While this is not necessary when the DNA plug-in has been configured to provide unique numbers for a single attribute, it is necessary for multiple attributes. There may be instances where an entry does not allow each type of attribute defined for the range, or, more important, an entry allows all of the attributes types defined, but only a subset of the attributes require the unique value.

### Example 7.6. DNA and Unique Bank Account Numbers

Example Bank wants to use the same unique number for a customer's **primaryAccount** and **customerId** attributes. The Example Bank administrator configured the DNA Plug-in to assign unique values for both attributes from the same range.

The bank also wants to assign numbers for secondary accounts from the same range as the customer ID and primary account numbers, but these numbers cannot be the same as the primary account numbers. The Example Bank administrator configures the DNA Plug-in to also manage the

**secondaryAccount** attribute, but will only add the **secondaryAccount** attribute to an entry *after* the entry is created and the **primaryAccount** and **customerID** attributes are assigned. This ensures that **primaryAccount** and **customerID** share the same unique number, and any **secondaryAccount** numbers are entirely unique but still from the same range of numbers.

## 7.4.2. Looking at the DNA Plug-in Syntax

The DNA Plug-in itself is a container entry, similar to the Password Storage Schemes Plug-in. Each DNA entry underneath the DNA Plug-in entry defines a new managed range for the DNA Plug-in.

To set new managed ranges for the DNA Plug-in, create entries beneath the container entry.

The most basic configuration is to set up distributed numeric assignments on a single server, meaning the ranges will not be shared or transferred between servers. A basic DNA configuration entry defines four things:

- The attribute that value is being managed, set in the **dnaType** attribute
- The entry DN to use as the base to search for entries, set in the **dnaScope** attribute
- The search filter to use to identify entries to manage, set in the **dnaFilter** attribute
- The next available value to assign, set in the **dnaNextValue** attribute (after the entry is created, this is handled by the plug-in)

For a list of attributes supported in the **cn=DNA\_config\_entry,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config** entry, see the [Red Hat Directory Server Configuration, Command, and File Reference](#).

To configure distributed numeric assignment on a single server for a single attribute type:

```
dn: cn=Account UIDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
objectClass: top
objectClass: dnaPluginConfig
cn: Account UIDs
dnatype: uidNumber
dnafilter: (objectclass=posixAccount)
dnascope: ou=people,dc=example,dc=com
dnaNextValue: 1
```

If multiple suppliers are configured for distributed numeric assignments, then the entry must contain the required information to transfer ranges:

- The maximum number that the server can assign; this sets the upward bound for the range, which is logically required when multiple servers are assigning numbers. This is set in the **dna.MaxValue** attribute.
- The threshold where the range is low enough to trigger a range transfer, set in the **dna.Threshold** attribute. If this is not set, the default value is **1**.
- A timeout period so that the server does not hang waiting for a transfer, set in the **dna.RangeRequestTimeout** attribute. If this is not set, the default value is **10**, meaning 10 seconds.

- A configuration entry DN which is shared among all supplier servers, which stores the range information for each supplier, set in the ***dnaSharedCfgDN*** attribute.

The specific number range which could be assigned by the server is defined in the ***dnaNextRange*** attribute. This shows the next available range for transfer and is managed automatically by the plug-in, as ranges are assigned or used by the server. This range is just "on deck." It has not yet been assigned to another server and is still available for its local Directory Server to use.

```
dn: cn=Account UIDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
objectClass: top
objectClass: dnaPluginConfig
cn: Account UIDs
dnatype: uidNumber
dnafilter: (objectclass=posixAccount)
dnascope: ou=People,dc=example,dc=com
dnanextvalue: 1
dnaMaxValue: 1300
dnasharedcfgdn: cn=Account UIDs,ou=Ranges,dc=example,dc=com
dnathreshold: 100
dnaRangeRequestTimeout: 60
dnaNextRange: 1301-2301
```

The ***dnaNextRange*** attribute should be set explicitly only if a separate, specific range has to be assigned to other servers. Any range set in the ***dnaNextRange*** attribute must be unique from the available range for the other servers to avoid duplication. If there is no request from the other servers and the server where ***dnaNextRange*** is set explicitly has reached its set ***dna.MaxValue***, the next set of values (part of the ***dnaNextRange***) is allocated from this deck.

The ***dnaNextRange*** allocation is also limited by the ***dnaThreshold*** attribute that is set in the DNA configuration. Any range allocated to another server for ***dnaNextRange*** cannot violate the threshold for the server, even if the range is available on the deck of ***dnaNextRange***.



#### NOTE

If the ***dnaNextRange*** attribute is handled internally if it is not set explicitly. When it is handled automatically, the ***dna.MaxValue*** attribute serves as upper limit for the next range.

Each supplier keeps a track of its current range in a separate configuration entry which contains information about its range and its connection settings. This entry is a child of the location in ***dnasharedcfgdn***. The configuration entry is replicated to all of the other suppliers, so each supplier can check that configuration to find a server to contact for a new range. For example:

```
dn: dnaHostname=ldap1.example.com+dnaportNum=389,cn=Account
UIDs,ou=Ranges,dc=example,dc=com
objectClass: dnaSharedConfig
objectClass: top
dnahostname: ldap1.example.com
dnaportNum: 389
dnasecureportnum: 636
dnaremainingvalues: 1000
```

### 7.4.3. Configuring Unique Number Assignments

The unique number distribution is configured by creating different instances of the DNA Plug-in.

#### 7.4.3.1. Creating a New Instance of the DNA Plug-in

To use the DNA with multiple configurations, create a new instance of the plug-in for each configuration.



##### NOTE

You can create new instances of the plug-in only by using the command line. However, you can edit the settings using both the command line and the web console.

To create and enabling a new instance of the plug-in:

1. For example, to create a new instance of the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin dna config "Account UIDs" add --type uidNumber --filter "(objectclass=posixAccount)" --scope ou=People,dc=example,dc=com --next-value 1 --max-value 1300 --shared-config-entry "cn=Account UIDs,ou=Ranges,dc=example,dc=com" --threshold 100 --range-request-timeout 60 --dnaMagicRegen: magic
```

2. Enable the DNA plug-in. For details, see [Section 1.10.3, “Enabling and Disabling Plug-ins”](#).

#### 7.4.3.2. Configuring Unique Number Assignments Using the Command Line



##### NOTE

Any attribute which has a unique number assigned to it must have an equality index set for it. The server must perform a sorted search, internally, to see if the **dnaNextvalue** is already taken, which requires an equality index on an integer attribute, with the proper ordering matching rule.

Creating indexes is described in [Section 13.2, “Creating Standard Indexes”](#).



##### NOTE

Set up the DNA Plug-in on every supplier server, and be careful not to overlap the number range values.

1. Create a new instance of the plug-in. See [Section 7.4.3.1, “Creating a New Instance of the DNA Plug-in”](#).
2. Create the shared container entry in the replicated subtree:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
dn: ou=Ranges,dc=example,dc=com  
changetype: add  
objectclass: top  
objectclass: extensibleObject  
objectclass: organizationalUnit  
ou: Ranges
```

```
-dn: cn=Account UIDs,ou=Ranges,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
cn: Account UIDs
```

3. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 7.4.3.3. Configuring Unique Number Assignments Using the Web Console

To enable and configure the **DNA** plug-in using the web console:

1. Create a new instance of the plug-in. See [Section 7.4.3.1, “Creating a New Instance of the DNA Plug-in”](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. Open the **Plugins** menu.
5. Select the **DNA** plug-in.
6. Change the status to **ON** to enable the plug-in.
7. Click **Add Config**.
8. Fill the fields, and enable the config.
9. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 7.4.4. Distributed Number Assignment Plug-in Performance Notes

There can be thread locking issues as DNA configuration is changed dynamically, so that new operations which access the DNA configuration (such as a DNA task or additional changes to the DNA configuration) will access the old configuration because the thread with the new configuration has not yet been released. This can cause operations to use old configuration or simply cause operations to hang.

To avoid this, preserve an interval between dynamic DNA configuration changes of 35 seconds. This means have a sleep or delay between both DNA configuration changes and any directory entry changes which would trigger a DNA plug-in operation.

# CHAPTER 8. ORGANIZING AND GROUPING ENTRIES

Entries contained within the directory can be grouped in different ways to simplify the management of user accounts. Red Hat Directory Server supports a variety of methods for grouping entries and sharing attributes between entries. To take full advantage of the features offered by roles and class of service, determine the directory topology when planning the directory deployment.

## 8.1. USING GROUPS

Similar to the operating system, you can add users to groups in Directory Server. Groups work the other way around as roles. If you are using roles, the DN of the assigned role is stored in the ***nsRoleDN*** attribute in the user object. If you use groups, then the DN of the users who are members of this group are stored in ***member*** attributes in the group object. If you enabled the ***memberOf*** plug-in, then the groups the user is a member of, are additionally stored in ***memberOf*** attribute in the user object. With this plug-in enabled, groups additionally have the benefit of roles, that you can list the group memberships of a user, similar as when using roles. Additionally, groups are faster than roles.

For further details about using the ***memberOf*** plug-in, see [Section 8.1.4, “Listing Group Membership in User Entries”](#).

### 8.1.1. The Different Types of Groups

Creating both static and dynamic groups from the command line is a similar process. A group entry contains the group name, the type of group, and a members attribute.

There are several different options for the type of group; these are described in more detail in the [Red Hat Directory Server 10 Configuration, Command, and File Reference](#). The *type of group* in this case refers to the type of defining member attribute it has:

- **groupOfNames** (recommended) is a simple group, that allows any entry to be added. The attribute used to determine members for this is ***member***.
- **groupOfUniqueNames**, like **groupOfNames**, simply lists user DNs as members, but the members must be unique. This prevents users being added more than once as a group member, which is one way of preventing self-referential group memberships. The attribute used to determine members for this is ***uniqueMember***.
- **groupOfURLs** uses a list of LDAP URLs to filter and generate its membership list. This object class is required for any dynamic group and can be used in conjunction with **groupOfNames** and **groupOfUniqueNames**.
- **groupOfCertificates** is similar to **groupOfURLs** in that it uses an LDAP filter to search for and identify certificates (or, really, certificate names) to identify group members. This is useful for group-based access control, since the group can be given special access permissions. The attribute used to determine members for this is ***memberCertificate***.

The following table shows the default attributes for groups:

**Table 8.1. Dynamic and Static Group Schema**

Type of Group	Group Object Classes	Member Attributes
Static	<b><i>groupOfNames</i></b> <sup>[a]</sup>	<b><i>member</i></b>

Type of Group	Group Object Classes	Member Attributes
Dynamic	<i>groupOfUniqueNames</i> <sup>[a]</sup>	<i>uniqueMember</i>
	<i>groupOfURLs</i>	<i>memberURL</i>
	<i>groupOfCertificates</i>	<i>memberCertificate</i>

[a] If this object class is used together with one of the dynamic object classes, the group becomes dynamic.

The following two examples show a static and a dynamic group entry:

### Example 8.1. A Static Group Entry

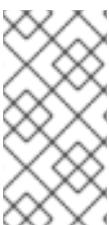
A static group entry lists the specific members of the group. For example:

```
objectClass: top
objectClass: groupOfUniqueNames
cn: static group
description: Example static group.
uniqueMember: uid=mwhite,ou=People,dc=example,dc=com
uniqueMember: uid=awhite,ou=People,dc=example,dc=com
```

### Example 8.2. A Dynamic Group Entry

A dynamic group uses at least one LDAP URL to identify entries belonging to the group and can specify multiple LDAP URLs or, if used with another group object class like **groupOfUniqueNames**, can explicitly list some group members along with the dynamic LDAP URL. For example:

```
objectClass: top
objectClass: groupOfUniqueNames
objectClass: groupOfURLs
cn: dynamic group
description: Example dynamic group.
memberURL: ldap://dc=example,dc=com??sub?(&(objectclass=person)(cn=*sen*))
```



#### NOTE

The **memberOf** plug-in does not support dynamically generated group memberships. If you set the **memberURL** attribute instead of listing the group members in an attribute, the **memberOf** plug-in does not add the **memberOf** attribute to the user objects that match the filter.

#### 8.1.2. Creating a Static Group

Directory Server only supports creating static groups using the command line.

### 8.1.2.1. Creating a Static Group Using the Command Line

This section describes how to create the different types of static groups using the command line.

For details about the different static groups, see [Section 8.1.1, “The Different Types of Groups”](#).

#### Creating a Static Group with the `groupOfNames` Object Class

The `dsidm` utility creates static groups in the `cn=Groups` entry in the specified base DN.

For example, to create the static `example_group` group with the `groupOfNames` object class in the `cn=Groups,dc=example,dc=com` entry

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" group
create --cn "example_group"
```

#### Creating a Static Group with the `groupOfUniqueNames` Object Class

To create a static group with the `groupOfUniqueNames` object class, use the `ldapmodify` utility to add the entry.

For example, to create the static `example_group` group with the `groupOfUniqueNames` object class in the `cn=Groups,dc=example,dc=com` entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_group,cn=Groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfUniqueNames
cn: example_group
description: Example static group with unique members
```

### 8.1.3. Creating a Dynamic Group

Directory Server only supports creating dynamic groups using the command line.

#### 8.1.3.1. Creating a Dynamic Group Using the Command Line

This section describes how to create the different types of dynamic groups using the command line.

For details about the different dynamic groups, see [Section 8.1.1, “The Different Types of Groups”](#).

#### Creating a Dynamic Group with the `groupOfURLs` Object Class

For example, to create the dynamic `example_group` group with the `groupOfURLs` object class in the `cn=Groups,dc=example,dc=com` entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_group,cn=Groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfURLs
cn: example_group
description: Example dynamic group for user entries
memberURL: ldap:///dc=example,dc=com??sub?((&(objectclass=person)(cn=*sen*)))
```

#### Creating a Dynamic Group with the `groupOfCertificates` Object Class

For example, to create the dynamic **example\_group** group with the **groupOfCertificates** object class in the **cn=Groups,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_group,cn=Groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfURLs
cn: example_group
description: Example dynamic group for certificate entries
memberCertificate: ...
```

## 8.1.4. Listing Group Membership in User Entries

The entries which belong to a group are defined, in some way, in the group entry itself. This makes it very easy to look at a group and see its members and to manage group membership centrally. However, there is no good way to find out what groups a single user belongs to. There is nothing in a user entry which indicates its memberships, as there are with roles.

The MemberOf Plug-in correlates group membership lists to the corresponding user entries.

The MemberOf Plug-in analyzes the member attribute in a group entry and automatically writes a corresponding **memberOf** attribute in the member's entry. (By default, this checks the **member** attribute, but multiple attribute instances can be used to support multiple different group types.)

As membership changes, the plug-in updates the **memberOf** attributes on the user entries. The MemberOf Plug-in provides a way to view the groups to which a user belongs simply by looking at the entry, including nested group membership. It can be very difficult to backtrack memberships through nested groups, but the MemberOf Plug-in shows memberships for all groups, direct and indirect.

The MemberOf Plug-in manages member attributes for static groups, not dynamic groups or circular groups.

### 8.1.4.1. Considerations When Using the **memberOf** Plug-in

This section describes important considerations when you want to use the **memberOf** plug-in.

#### Using the **memberOf** Plug-in in a Replication Topology

There are two approaches to manage the **memberOf** attribute in a replication topology:

- Enable the **memberOf** plug-in on all supplier and read-only replica servers in the topology. In this case, you must exclude the **memberOf** attribute from replication in all replication agreements. For details about about excluding attributes, see [Section 15.1.7, “Replicating a Subset of Attributes with Fractional Replication”](#).
- Enable the **memberOf** plug-in only on all supplier servers in the topology. For this:
  - You must disable replication of the **memberOf** attribute to all write-enabled suppliers in the replication agreement. For details about about excluding attributes, see [Section 15.1.7, “Replicating a Subset of Attributes with Fractional Replication”](#) .
  - You must Enable replication of the **memberOf** attribute to all read-only replicas in their replication agreement.
  - You must not enable the **memberOf** plug-in on read-only replicas.

## Using the **memberOf** plug-in With Distributed Databases

As described in [Section 2.2.1, "Creating Databases"](#), you can store sub-trees of your directory in individual databases. By default, the **memberOf** plug-in only updates user entries which are stored within the same database as the group. To enable the plug-in to also update users in different databases as the group, you must set the **memberOfAllBackends** parameter to **on**. See [Section 8.1.4.5.2, "Configuring the MemberOf Plug-in on Each Server Using the Web Console"](#).

### 8.1.4.2. Required Object Classes by the **memberOf** Plug-In

The **memberOf** plug-in By default, the **memberOf** plug-in will add the **MemberOf** object class to objects to provide the **memberOf** attribute. This object class is safe to add to any object for this purpose, and no further action is required to enable this plug-in to operate correctly. Alternatively, you can create user objects that contain the **inetUser** or **inetAdmin**, object class. Both object classes support the **memberOf** attribute as well.

To configure nested groups, the group must use the **extensibleObject** object class.



#### NOTE

If directory entries do not contain an object class that supports the required attributes, operations fail with the following error:

LDAP: error code 65 - Object Class Violation

### 8.1.4.3. The MemberOf Plug-in Syntax

The MemberOf Plug-in instance defines two attributes, one for the group member attribute to poll (**memberOfGroupAttr**) and the other for the attribute to create and manage in the member's user entry (**memberOfAttr**).

The **memberOfGroupAttr** attribute is multi-valued. Because different types of groups use different member attributes, using multiple **memberOfGroupAttr** attributes allows the plug-in to manage multiple types of groups.

The plug-in instance also gives the plug-in path and function to identify the MemberOf Plug-in and contains a state setting to enable the plug-in, both of which are required for all plug-ins. The default MemberOf Plug-in is shown in [Example 8.3, "Default MemberOf Plug-in Entry"](#).

#### Example 8.3. Default MemberOf Plug-in Entry

```
dn: cn=memberOf Plugin,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: MemberOf Plugin
nsslapd-pluginPath: libmemberof-plugin
nsslapd-pluginInitfunc: memberof_postop_init
nsslapd-pluginType: postoperation
nsslapd-pluginEnabled: on
nsslapd-plugindepends-on-type: database
memberOfGroupAttr: member
memberOfGroupAttr: uniqueMember
memberOfAttr: memberOf
```

```
memberOfAllBackends: on
nsslapd-pluginId: memberOf
nsslapd-pluginVersion: X.Y.Z
nsslapd-pluginVendor: Red Hat, Inc.
nsslapd-pluginDescription: memberOf plugin
```

For details about the parameters used in the example and other parameters you can set, see the [MemberOf Plug-in Attributes](#) section in the *Red Hat Directory Server Command, Configuration, and File Reference*.



#### NOTE

To maintain backwards compatibility with older versions of Directory Server, which only allowed a single member attribute (by default, **member**), it may be necessary to include the **memberOf** group attribute or whatever previous member attribute was used, in addition any new member attributes used in the plug-in configuration.

```
memberOfGroupAttr: member
memberOfGroupAttr: uniqueMember
```

#### 8.1.4.4. Enabling the MemberOf Plug-in

This section describes how to enable the **memberOf** plug-in.

##### 8.1.4.4.1. Enabling the MemberOf Plug-in Using the Command Line

Enable the **memberOf** plug-in using the command line:

1. Use the **dsconf** utility to enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

##### 8.1.4.4.2. Enabling the MemberOf Plug-in Using the Web Console

Enable the **memberOf** plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **memberOf** plug-in.

5. Change the status to **ON** to enable the plug-in.
6. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 8.1.4.5. Configuring the MemberOf Plug-in on Each Server

If you do not want to replicate the configuration of the **MemberOf** plug-in, configure the plug-in manually on each server.

##### 8.1.4.5.1. Configuring the MemberOf Plug-in on Each Server Using the Command Line

To configure the **MemberOf** plug-in using the command line:

1. Enable the plug-in. See [Section 8.1.4.4.1, “Enabling the MemberOf Plug-in Using the Command Line”](#).
2. To retrieve members of a group from a different attribute than **member**, which is the default, set the **memberOfGroupAttr** parameter to the respective attribute name.

For example, to read group members from **uniqueMember** attributes, replace the current value of **memberOfGroupAttr**.

- a. Optionally, display the attribute that is currently configured:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof show
...
memberOfgroupattr: member
...
```

The command displays that currently only the **member** attribute is configured to retrieve members of a group.

- b. Remove all attributes from the configuration that currently set:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --
groupattr delete
Successfully changed the cn=memberOf Plugin,cn=plugins,cn=config
```



##### NOTE

It is not possible to remove a specific group attribute.

- c. Add the **uniqueMember** attribute to the configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --
groupattr uniqueMember
successfully added memberOfGroupAttr value "uniqueMember"
```

To set multiple attributes, pass them all to the **--groupattr** parameter. For example:



```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --
groupattr member uniqueMember ...
```

3. By default, the **MemberOf** plug-in adds the **memberOf** attribute to user entries. To use a different attribute, set the name of the attribute in the **memberOfAttr** parameter.

For example, to add the **customMemberOf** attribute to user records, replace the current value of **memberOfAttr**.

- a. Optionally, display the attribute that is currently configured:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof show
...
memberOfattr: memberOf
...
```

- b. Configure the **MemberOf** plug-in to add the **customMemberOf** attribute to user entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --
attr customMemberOf
memberOfAttr set to "customMemberOf"
```



#### NOTE

You can only set this parameter to an attribute that supports DN syntax.

4. In an environment that uses distributed databases, you can configure the plug-in to search user entries in all databases instead of only the local database:

```
dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --
allbackends on
memberOfAllBackends enabled successfully
```

5. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 8.1.4.5.2. Configuring the MemberOf Plug-in on Each Server Using the Web Console

To configure the **MemberOf** plug-in using the command line:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **memberOf** plug-in.

5. Change the status to **ON** to enable the plug-in.
6. Fill the fields to configure the plug-in. For example, to configure that the plug-in adds the **customMemberOf** attribute to user entries if the **uniqueMember** attribute is added to a group:

**MemberOf**

Status **ON**

Attribute	uniqueMember
Group Attribute	customMemberOf
Entry Scope	<input checked="" type="checkbox"/> All Backends
Entry Scope Exclude Subtree	<input type="checkbox"/> Skip Nested
Shared Config Entry	Type an objectClass...
<b>Manage</b>	
<b>Run Fixup Task</b>	

7. Click **Save**.
8. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 8.1.4.6. Using the MemberOf Plug-in Shared Configuration

By default, the configuration of the **MemberOf** plug-in is stored on each server. Using the shared configuration feature of the plug-in, the configuration can be stored outside of the **cn=config** suffix and replicated. Administrators can use the same settings without configuring the plug-in manually on each server.

1. Enable the plug-in. See [Section 8.1.4.4, “Enabling the MemberOf Plug-in”](#).
2. Add the shared configuration entry for the **MemberOf** plug-in. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof config-entry
add "cn=shared_MemberOf_config,dc=example,dc=com" --groupattr "member" --attr
"memberOf"
```

This automatically enables the shared configuration entry on the server on which you ran the command.

3. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

4. On all other servers in the replication topology that should use the shared configuration, enable the shared configuration:

a. Enable the plug-in. See [Section 8.1.4.4, “Enabling the MemberOf Plug-in”](#).

b. Set the DN that stores the shared configuration. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --config-entry cn=shared_MemberOf_config,dc=example,dc=com
```

c. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.



### IMPORTANT

After enabling the shared configuration, the plug-in ignores all parameters set in the **cn=MemberOf Plugin,cn=plugins,cn=config** plug-in entry and only uses settings from the shared configuration entry.

#### 8.1.4.7. Setting the Scope of the MemberOf Plug-in

If you configured several back ends or multiple-nested suffixes, you can use the **memberOfEntryScope** and **memberOfEntryScopeExcludeSubtree** parameters to set what suffixes the **MemberOf** plug-in works on.

If you add a user to a group, the **MemberOf** plug-in only adds the **memberOf** attribute to the group if both the user and the group are in the plug-in's scope. For example, to configure the **MemberOf** plug-in to work on all entries in **dc=example,dc=com**, but to exclude entries in **ou=private,dc=example,dc=com**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --scope "dc=example,com"
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --exclude "dc=group,dc=example,com"
```

If you moved a user entry out of the scope by using the **--scope DN** parameter:

- The membership attribute, such as **member**, is updated in the group entry to remove the user DN value.
- The **memberOf** attribute is updated in the user entry to remove the group DN value.



### NOTE

The value set in the **--exclude** parameter has a higher priority than values set in **--scope**. If the scopes set in both parameters overlap, the **MemberOf** plug-in only works on the non-overlapping directory entries.

#### 8.1.4.8. Regenerating **memberOf** Values

The **MemberOf** plug-in automatically manages **memberOf** attributes on group member entries, based on the configuration in the group entry itself. However, the **memberOf** attribute can be manually edited

in a user entry or new entries can be imported or replicated to the server that have a **memberOf** attribute already set. These situations create inconsistencies between the **memberOf** configuration managed by the server plug-in and the actual memberships defined in an entry.

For example, to regenerate the **memberOf** values in **dc=example,dc=com** entry and subentries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof fixup -f "(|(objectclass=inetuser)(objectclass=inetadmin)(objectclass=nsmemberof))" "dc=example,dc=com"
Attempting to add task entry...
Successfully added task entry
```

The **-f filter** option is optional. Use the filter to regenerate the **memberOf** attributes in user entries matching the filter. If you do not specify a filter, the tasks regenerates the attributes in all entries containing the **inetUser**, **inetAdmin**, or **nsMemberOf** object class.



#### NOTE

Regeneration tasks run locally, even if the entries themselves are replicated. This means that **memberOf** attributes for entries on other servers are not updated until the updated entry is replicated.

### 8.1.5. Automatically Adding Entries to Specified Groups

- [Section 8.1.5.1, “Looking at the Structure of an Automembership Rule”](#)
- [Section 8.1.5.4, “Examples of Automembership Rules”](#)
- [Section 8.1.5.2, “Configuring Auto Membership Definitions”](#)

Group management can be a critical factor for managing directory data, especially for clients which use Directory Server data and organization or which use groups to apply functionality to entries. Groups make it easier to apply policies consistently and reliably across the directory. Password policies, access control lists, and other rules can all be based on group membership.

Being able to assign new entries to groups, automatically, at the time that an account is created ensures that the appropriate policies and functionality are immediately applied to those entries – without requiring administrator intervention.

Dynamic groups are one method of creating groups and assigning members automatically because any matching entry is automatically included in the group. For applying Directory Server policies and settings, this is sufficient. However, LDAP applications and clients commonly need a static and explicit list of group members in order to perform whatever operation is required. And all of the members in static groups have to be manually added to those groups.

The static group itself cannot search for members like a dynamic group, but there is a way to allow a static group to have members added to it automatically – *the Auto Membership Plug-in*.

Automembership essentially allows a static group to act like a dynamic group. Different automembership definitions create searches that are automatically run on all new directory entries. The automembership rules search for and identify matching entries – much like the dynamic search filters – and then explicitly add those entries as members to the static group.



## NOTE

By default, the ***autoMemberProcessModifyOps*** parameter in the **cn=Auto Membership Plugin,cn=plugins,cn=config** entry is set to **on**. With this setting, the Automembership plug-in also updates group memberships when an administrator moves a user to a different group by editing a user entry.

If you set ***autoMemberProcessModifyOps*** to **off**, Directory Server invokes the plug-in only when you add a group entry to the user, and you must manually run a fix-up task to update the group membership.

The Auto Membership Plug-in can target any type of object stored in the directory: users, machines and network devices, customer data, or other assets.



## NOTE

The Auto Membership Plug-in adds a new member to an existing group based on defined criteria. It does not create a group for the new entry.

To create a corresponding group entry when a new entry of a certain type is created, use the Managed Entries Plug-in. This is covered in [Section 8.3, “Automatically Creating Dual Entries”](#).

### 8.1.5.1. Looking at the Structure of an Automembership Rule

The Auto Membership Plug-in itself is a container entry in **cn=plugins,cn=config**. Group assignments are defined through child entries.

#### 8.1.5.1.1. The Automembership Configuration Entry

Automembership assignments are created through a main definition entry, a child of the Auto Membership Plug-in entry. Each definition entry defines three elements:

- An LDAP search to identify entries, including both a search scope and a search filter (***autoMemberScope*** and ***autoMemberFilter***)
- A default group to which to add the member entries (***autoMemberDefaultGroup***)
- The member entry format, which is the attribute in the group entry, such as ***member***, and the attribute value, such as ***dn (autoMemberGroupingAttr)***

The definition is the basic configuration for an automember rule. It identifies all of the required information: what a matching member entry looks like and a group for that member to belong to.

For example, this definition assigns all users with the object class set to **ntUser** to the **cn=windows-users** group:

```
dn: cn=Windows Users,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
autoMemberScope: ou=People,dc=example,dc=com
autoMemberFilter: objectclass=ntUser
autoMemberDefaultGroup: cn=windows-group,cn=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

For details about the attributes used in the example and other attributes you can set in this entry, see the **cn=Auto Membership Plugin,cn=plugins,cn=config** entry description in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

#### 8.1.5.1.2. Additional Regular Expression Entries

For something like a users group, where more than likely all matching entries should be added as members, a simple definition is sufficient. However, there can be instances where entries that match the LDAP search filter should be added to different groups, depending on the value of some other attribute. For example, machines may need to be added to different groups depending on their IP address or physical location; users may need to be in different groups depending on their employee ID number.

The automember definition can use regular expressions to provide additional conditions on what entries to include or exclude from a group, and then a new, specific group to add those selected entries to.

For example, an automember definition sets all machines to be added to a generic host group.

##### Example 8.4. Automember Definition for a Host Group

```
dn: cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: dc=example,dc=com
autoMemberFilter: objectclass=ipHost
autoMemberDefaultGroup: cn=systems,cn=hostgroups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

A regular expression rule is added so that any machine with a fully-qualified domain name within a given range is added to a web server group.

##### Example 8.5. Regular Expression Condition for a Web Server Group

```
dn: cn=webservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group for webservers
cn: webservers
autoMemberTargetGroup: cn=webservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^www\.web[0-9]+\!.example\.com
```

So, any host machine added with a fully-qualified domain name that matches the expression **^www\.web[0-9]+\!.example\.com**, such as **www.web1.example.com**, is added to the **cn=webservers** group, defined for that exact regular expression. Any other machine entry, which matches the LDAP filter **objectclass=ipHost** but with a different type of fully-qualified domain name, is added to the general host group, **cn=systems**, defined in the main definition entry.

The group in the definition, then, is a fallback for entries which match the general definition, but do not meet the conditions in the regular expression rule.

Regular expression rules are child entries of the automember definition.

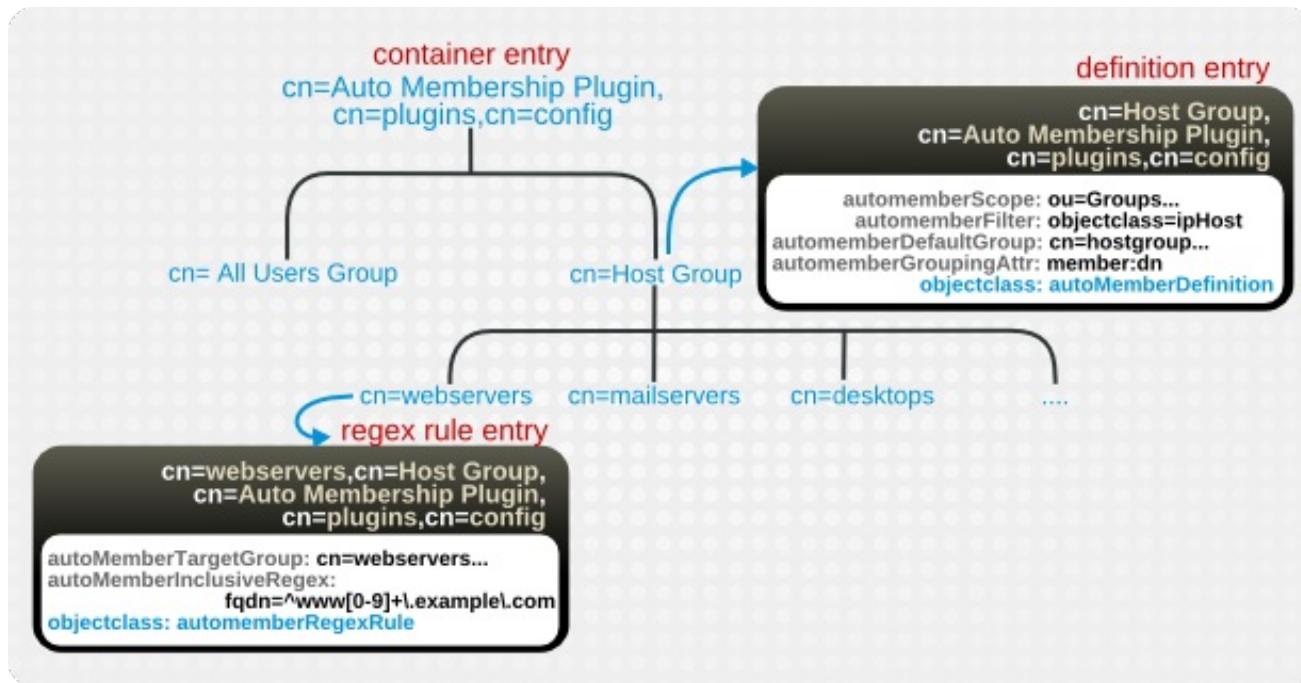


Figure 8.1. Regular Expression Conditions

Each rule can include multiple inclusion and exclusion expressions. (Exclusions are evaluated first.) If an entry matches any inclusion rule, it is added to the group.

There can be only one target group given for the regular expression rule.

Table 8.2. Regular Expression Condition Attributes

Attribute	Description
<code>autoMemberRegexRule</code> (required object class)	Identifies the entry as a regular expression rule. This entry must be a child of an automember definition ( <b>objectclass: autoMemberDefinition</b> ).
<code>autoMemberInclusiveRegex</code>	<p>Sets a regular expression to use to identify entries to include. Only matching entries are added to the group. Multiple regular expressions could be used, and if an entry matches any one of those expressions, it is included in the group.</p> <p>The format of the expression is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see the <b>pcresyntax(3)</b> man page.</p> <p>This is a multi-valued attribute.</p>

Attribute	Description
autoMemberExclusiveRegex	<p>Sets a regular expression to use to identify entries to exclude. If an entry matches the exclusion condition, then it is <i>not</i> included in the group. Multiple regular expressions could be used, and if an entry matches any one of those expressions, it is excluded in the group. The format of the expression is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see the <b>pcresyntax(3)</b> man page.</p> <p>This is a multi-valued attribute.</p> <p> <b>NOTE</b></p> <p>Exclude conditions are evaluated first and take precedence over include conditions.</p>
autoMemberTargetGroup	Sets which group to add the entry to as a member, if it meets the regular expression conditions.

### 8.1.5.2. Configuring Auto Membership Definitions

To use the **Auto Membership** plug-in, create definitions for the plug-in.

#### 8.1.5.2.1. Configuring Auto Membership Definitions Using the Command Line

To create **Auto Membership** definitions using the command line:

1. Enable the **Auto Membership** plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin automember enable
Enabled Auto Membership Plugin
```

2. Create a **Auto Membership** definition. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin automember
definition definition_name add --default-group "cn=windows-
group,cn=groups,dc=example,dc=com" --scope "ou=People,dc=example,dc=com" --filter
"objectclass=ntUser" --grouping-attr "member:dn"
Automember definition created successfully!
```

3. Optionally, you can set further parameters in an **Auto Membership** definition, for example, to use regular expressions to identify entries to include. Use the **ldapmodify** utility to add or update these parameters in the **cn=definition\_name,cn=Auto Membership Plugin,cn=plugins,cn=config** entry. For parameters you can set, see **cn=Auto Membership Plugin,cn=plugins,cn=config** entry description in the *Red Hat Directory Server Configuration, Command, and File Reference*.

4. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 8.1.5.2.2. Configuring Auto Membership Definitions Using the Web Console

To create **Auto Membership** definitions using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Plugins** menu.
4. Select the **Auto Membership** plug-in.
5. Change the status to **ON** to enable the plug-in.
6. Click **Add Definition**.
7. Fill the fields. For example:

Config Name	Exclusive Regex	Inclusive Regex	Target Group
6 ^ per page	1-0 of 0	< <	1 of 0 > >

8. Optionally, add a regular expression filter.

9. Click **Save**.
10. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 8.1.5.3. Updating Existing Entries to apply Auto Membership Definitions

By default, the ***autoMemberProcessModifyOps*** parameter in the **cn=Auto Membership Plugin,cn=plugins,cn=config** entry is enabled. With this setting, the **Automembership** plug-in also updates group memberships when an administrator moves a user to a different group by editing a user entry. However, if you set ***autoMemberProcessModifyOps*** to **off**, you must manually run a fix-up task when you added new entries to the directory or changed existing entries.

To create the task entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin automember fixup -f "filter" -s scope
```

When the task is completed, the entry is removed from the directory configuration.

### 8.1.5.4. Examples of Automembership Rules

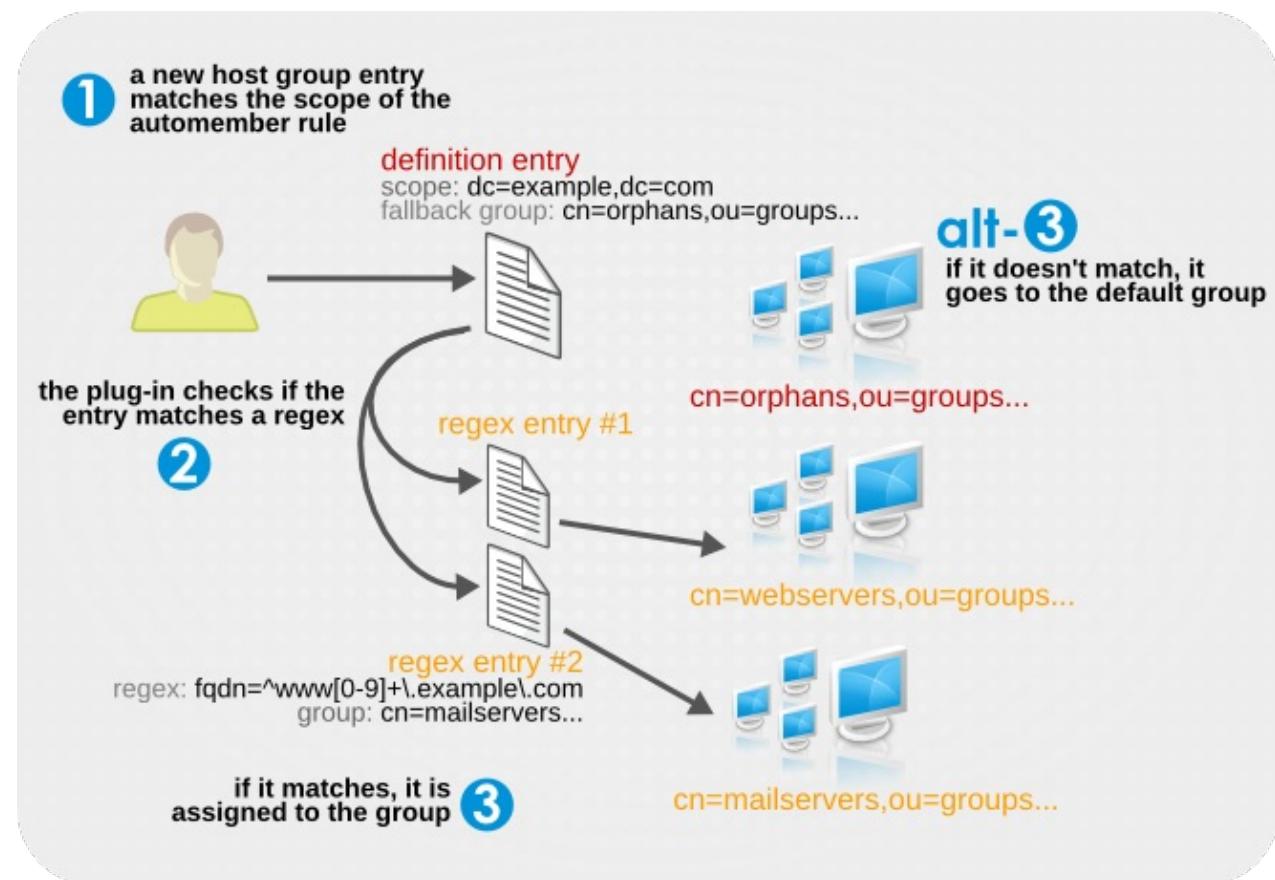
Automembership rules are usually going to applied to users and to machines (although they can be applied to any type of entry). There are a handful of examples that may be useful in planning automembership rules:

- Different host groups based on IP address
- Windows user groups
- Different user groups based on employee ID

#### Example 8.6. Host Groups by IP Address

The automember rule first defines the scope and target of the rule. The example in [Section 8.1.5.1.2, “Additional Regular Expression Entries”](#) uses the configuration group to define the fallback group and a regular expression entry to sort out matching entries.

The scope is used to find all host entries. The plug-in then iterates through the regular expression entries. If an entry matches an inclusive regular expression, then it is added to that host group. If it does not match any group, it is added to the default group.



The actual plug-in configuration entries are configured like this, for the definition entry and two regular expression entries to filter hosts into a web servers group or a mail servers group.

#### *configuration entry*

```
dn: cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: dc=example,dc=com
autoMemberFilter: objectclass=bootableDevice
autoMemberDefaultGroup: cn=orphans,cn=hostgroups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

#### *regex entry #1*

```
dn: cn=webservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group placement for webservers
cn: webservers
autoMemberTargetGroup: cn=webservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^www[0-9]+\example\.com
autoMemberInclusiveRegex: fqdn=^web[0-9]+\example\.com
autoMemberExclusiveRegex: fqdn=^www13\.example\.com
autoMemberExclusiveRegex: fqdn=^web13\.example\.com
```

#### *regex entry #2*

```
dn: cn=mailservers,cn=Hostgroups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group placement for mailservers
cn: mailservers
autoMemberTargetGroup: cn=mailservers,cn=hostgroups,dc=example,dc=com
autoMemberInclusiveRegex: fqdn=^mail[0-9]+\example\.com
```

```
autoMemberInclusiveRegex: fqdn=^smtp[0-9]+\example\com
autoMemberExclusiveRegex: fqdn=^mail13\example\com
autoMemberExclusiveRegex: fqdn=^smtp13\example\com
```

### Example 8.7. Windows User Group

The basic users group shown in [Section 8.1.5.1.1, “The Automembership Configuration Entry”](#) uses the **posixAccount** attribute to identify all new users. All new users created within Directory Server are created with the **posixAccount** attribute, so that is a safe catch-all for new Directory Server users. However, when user accounts are synchronized over from the Windows domain to the Directory Server, the Windows user accounts are created *without* the **posixAccount** attribute.

Windows users are identified by the **ntUser** attribute. The basic, all-users group rule can be modified to target Windows users specifically, which can then be added to the default all-users group or to a Windows-specific group.

```
dn: cn=Windows Users,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
autoMemberScope: dc=example,dc=com
autoMemberFilter: objectclass=ntUser
autoMemberDefaultGroup: cn=Windows Users,cn=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn
```

### Example 8.8. User Groups by Employee Type

The Auto Membership Plug-in can work on custom attributes, which can be useful for entries which are managed by other applications. For example, a human resources application may create and then reference users based on the employee type, in a custom **employeeType** attribute.

Much like [Example 8.6, “Host Groups by IP Address”](#), the user type rule uses two regular expression filters to sort full time and temporary employees, only this example uses an explicit value rather than a true regular expression. For other attributes, it may be more appropriate to use a regular expression, like basing the filter on an employee ID number range.

```
configuration entry
dn: cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberDefinition
cn: Hostgroups
autoMemberScope: ou=employees,ou=people,dc=example,dc=com
autoMemberFilter: objectclass=inetorgperson
autoMemberDefaultGroup: cn=general,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberGroupingAttr: member:dn

regex entry #1
dn: cn=full time,cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectclass: autoMemberRegexRule
description: Group for full time employees
cn: full time
autoMemberTargetGroup: cn=full time,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberInclusiveRegex: employeeType=full

regex entry #2
```

```

dn: cn=temporary,cn=Employee groups,cn=Auto Membership Plugin,cn=plugins,cn=config
objectClass: autoMemberRegexRule
description: Group placement for interns, contractors, and seasonal employees
cn: temporary
autoMemberTargetGroup: cn=temporary,cn=employee groups,ou=groups,dc=example,dc=com
autoMemberInclusiveRegex: employeeType=intern
autoMemberInclusiveRegex: employeeType=contractor
autoMemberInclusiveRegex: employeeType=seasonal

```

### 8.1.5.5. Testing Automembership Definitions

Because each instance of the Auto Member Plug-in is a set of related-but-separate entries for the definition and regular expression, it can be difficult to see exactly how users are going to be mapped to groups. This becomes even more difficult when there are multiple rules which target different subsets of users.

There are two dry-run tasks which can be useful to determine whether all of the different Auto Member Plug-in definitions are assigning groups properly as designed.

#### Testing with Existing Entries

**cn=automember export updates** runs against *existing entries* in the directory and exports the results of what users would have been added to what groups, based on the rules. This is useful for testing existing rules against existing users to see how your real deployment are performing.

This task requires the same information as the **cn=automember rebuild membership** task – the base DN to search, search filter, and search scope – and has an additional parameter to specify an export LDIF file to record the proposed entry updates.

```

# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=test_export,cn=automember export updates,cn=tasks,cn=config
objectClass: top
objectClass: extensibleObject
cn: test_export
basedn: dc=example,dc=com
filter: (uid=*)
scope: sub
ldif: /tmp/automember-updates.ldif

```

#### Testing with an Import LDIF

**cn=automember map updates** takes *an import LDIF* of new users and then runs the new users against the current automembership rules. This can be very useful for testing a new rule, before applying it to (real) new or existing user entries.

This is called a map task because it maps or relates changes for proposed new entries to the existing rules.

This task only requires two attributes: the location of the input LDIF (which must contain at least some user entries) and an output LDIF file to which to write the proposed entry updates. Both the input and output LDIF files are absolute paths on the local machine.

For example, using **ldapmodify**:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
  
dn: cn=test_mapping, cn=automember map updates,cn=tasks,cn=config  
objectClass: top  
objectClass: extensibleObject  
cn: test_mapping  
ldif_in: /tmp/entries.ldif  
ldif_out: /tmp/automember-updates.ldif
```

## 8.2. USING ROLES

Roles are an entry grouping mechanism that unify the static and dynamic groups described in the previous sections. Roles are designed to be more efficient and easier to use for applications. For example, an application can get the list of roles of which an entry is a member by querying the entry itself, rather than selecting a group and browsing the members list of several groups.

### 8.2.1. About Roles

Red Hat has two kinds of groups. *Static groups* have a finite and defined list of members. *Dynamic groups* used filters to recognize which entries are members of the group, so the group membership is constantly changed as the entries which match the group filter change. (Both kinds of groups are described in [Section 8.1, “Using Groups”](#).)

Roles are a sort of hybrid group, behaving as both a static and a dynamic group. With a group, entries are added to a group entry as members. With a role, the role attribute is added to an entry and then that attribute is used to identify members in the role entry automatically.

Role *members* are entries that possess the role. Members can be specified either explicitly or dynamically. How role membership is specified depends upon the type of role. Directory Server supports three types of roles:

- *Managed roles* have an explicit enumerated list of members.
- *Filtered roles* are assigned entries to the role depending upon the attribute contained by each entry, specified in an LDAP filter. Entries that match the filter possess the role.
- *Nested roles* are roles that contain other roles.

Managed roles can do everything that can normally be done with static groups. The role members can be filtered using filtered roles, similarly to the filtering with dynamic groups. Roles are easier to use than groups, more flexible in their implementation, and reduce client complexity.

When a role is created, determine whether a user can add themselves or remove themselves from the role. See [Section 8.2.7, “Using Roles Securely”](#) for more information about roles and access control.



## NOTE

Evaluating roles is more resource-intensive for the Directory Server than evaluating groups because the server does the work for the client application. With roles, the client application can check role membership by searching for the ***nsRole*** attribute. The ***nsRole*** attribute is a computed attribute, which identifies to which roles an entry belongs; the ***nsRole*** attribute is not stored with the entry itself. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

Considerations for using roles are covered in the *Red Hat Directory Server Deployment Guide*.

### 8.2.2. Creating a Managed Role

Managed roles have an explicit enumerated list of members. Managed roles are added to entries by adding the ***nsRoleDN*** attribute to the entry.

#### 8.2.2.1. Creating Managed Roles through the Command Line

Roles inherit from the ***Idapsubentry*** object class, which is defined in the ITU X.509 standard. In addition, each managed role requires two object classes that inherit from the ***nsRoleDefinition*** object class:

- nsSimpleRoleDefinition
- nsManagedRoleDefinition

A managed role also allows an optional ***description*** attribute.

Members of a managed role have the ***nsRoleDN*** attribute in their entry.

This example creates a role which can be assigned to the marketing department.

1. Use ***ldapmodify*** with the ***-a*** option to add the managed role entry. The new entry must contain the ***nsManagedRoleDefinition*** object class, which in turn inherits from the ***LdapSubEntry***, ***nsRoleDefinition***, and ***nsSimpleRoleDefinition*** object classes.

```
dn: cn=Marketing,ou=people,dc=example,dc=com
objectclass: top
objectclass: LdapSubEntry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: Marketing
description: managed role for marketing staff
```

2. Assign the role to the marketing staff members, one by one, using ***ldapmodify***:

```
dn: cn=Bob,ou=people,dc=example,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
```

The ***nsRoleDN*** attribute in the entry indicates that the entry is a member of a managed role, ***cn=Marketing,ou=people,dc=example,dc=com***.

### 8.2.3. Creating a Filtered Role

Entries are assigned to a filtered role depending whether the entry possesses a specific attribute defined in the role. The role definition specifies an LDAP filter for the target attributes. Entries that match the filter possess (are members of) the role.

#### 8.2.3.1. Creating a Filtered Role through the Command Line

Roles inherit from the **Idapsubentry** object class, which is defined in the ITU X.509 standard. In addition, each filtered role requires two object classes that inherit from the **nsRoleDefinition** object class:

- nsComplexRoleDefinition
- nsFilteredRoleDefinition

A filtered role entry also requires the **nsRoleFilter** attribute to define the LDAP filter to determine role members. Optionally, the role can take a **description** attribute.

Members of a filtered role are entries that match the filter specified in the **nsRoleFilter** attribute.

This example creates a filtered role which is applied to all sales managers.

1. Run **Idapmodify** with the **-a** option to add a new entry.
2. Create the filtered role entry.

The role entry has the **nsFilteredRoleDefinition** object class, which inherits from the **LdapSubEntry**, **nsRoleDefinition**, and **nsComplexRoleDefinition** object classes.

The **nsRoleFilter** attribute sets a filter for ***o*** (organization) attributes that contain a value of **sales managers**.

```
dn: cn=SalesManagerFilter,ou=people,dc=example,dc=com
changetype: add
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: SalesManagerFilter
nsRoleFilter: o=sales managers
Description: filtered role for sales managers
```

The following entry matches the filter (possesses the ***o*** attribute with the value **sales managers**), and, therefore, it is a member of this filtered role automatically:

```
dn: cn=Pat Smith,ou=people,dc=example,dc=com
objectclass: person
cn: Pat
sn: Smith
userPassword: secret
o: sales managers
```

### 8.2.4. Creating a Nested Role

Nested roles are roles that contain other roles. Before it is possible to create a nested role, another role must exist. The roles nested within the nested role are specified using the **nsRoleDN** attribute.

#### 8.2.4.1. Creating Nested Role through the Command Line

Roles inherit from the **LDAPsubentry** object class, which is defined in the ITU X.509 standard. In addition, each nested role requires two object classes that inherit from the **nsRoleDefinition** object class:

- nsComplexRoleDefinition
- nsNestedRoleDefinition

A nested role entry also requires the **nsRoleDN** attribute to identify the roles to nest within the container role. Optionally, the role can take a **description** attribute.

Members of a nested role are members of the roles specified in the **nsRoleDN** attributes of the nested role definition entry.

This example creates a single role out of the managed marketing role and filtered sales manager role.

1. Run **ldapmodify** with the **-a** option to add a new entry.
2. Create the nested role entry. The nested role has four object classes:
  - **nsNestedRoleDefinition**
  - **LDAPsubentry** (inherited)
  - **nsRoleDefinition** (inherited)
  - **nsComplexRoleDefinition** (inherited)

The **nsRoleDN** attributes contain the DNs for both the marketing managed role and the sales managers filtered role.

```
dn: cn=MarketingSales,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsNestedRoleDefinition
cn: MarketingSales
nsRoleDN: cn=SalesManagerFilter,ou=people,dc=example,dc=com
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
```

Both of the users in the previous examples, Bob and Pat, are members of this new nested role.

#### 8.2.5. Viewing Roles for an Entry through the Command Line

Role assignments are not returned automatically through the command line.

The **nsRole** attribute is an operational attribute. In LDAP, operational attributes must be requested explicitly. They are not returned by default with the regular attributes in the schema of the entry. You can either explicitly request single operational attributes by listing them or use **+** to output all operational attributes for result objects. For example, this **ldapsearch** command returns the list of roles of which **uid=user\_name** is a member, in addition to the regular attributes for the entry:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(uid=user_name)"" \* nsRole

dn: uid=user_name,ou=people,dc=example,dc=com
...
nsRole: cn=Role for Managers,dc=example,dc=com
nsRole: cn=Role for Accounting,dc=example,dc=com
```

## 8.2.6. About Deleting Roles

Deleting a role deletes the role entry but does not delete the ***nsRoleDN*** attribute for each role member. To delete the ***nsRoleDN*** attribute for each role member, enable the Referential Integrity plug-in, and configure it to manage the ***nsRoleDN*** attribute. For more information on the Referential Integrity plug-in, see [Chapter 5, Maintaining Referential Integrity](#).

## 8.2.7. Using Roles Securely

Not every role is suitable for use in a security context. When creating a new role, consider how easily the role can be assigned to and removed from an entry. Sometimes it is appropriate for users to be able to add or remove themselves easily from a role. For example, if there is an interest group role called **Mountain Biking**, interested users should be able to add themselves or remove themselves easily.

However, it is inappropriate to have such open roles for some security situations. One potential security risk is inactivating user accounts by inactivating roles. Inactive roles have special ACIs defined for their suffix. If an administrator allows users to add and remove themselves from roles freely, then in some circumstance, they may be able to remove themselves from an inactive role to prevent their accounts from being locked.

For example, user A possesses the managed role, **MR**. The **MR** role has been locked using account inactivation. This means that user A cannot bind to the server because the ***nsAccountLock*** attribute is computed as **true** for that user. However, if user A was already bound to Directory Server and noticed that he is now locked through the **MR** role, the user can remove the ***nsRoleDN*** attribute from his entry and unlock himself if there are no ACIs preventing him.

To prevent users from removing the ***nsRoleDN*** attribute, use the following ACIs depending upon the type of role being used.

- *Managed roles.* For entries that are members of a managed role, use the following ACI to prevent users from unlocking themselves by removing the appropriate ***nsRoleDN***:

```
aci: (targetattr="nsRoleDN") (targattrfilters= add=nsRoleDN:(!
(nsRoleDN=cn=AdministratorRole,dc=example,dc=com)), del=nsRoleDN:(!
(nsRoleDN=cn=nsManagedDisabledRole,dc=example,dc=com))) (version3.0;acl "allow mod
of nsRoleDN by self but not to critical values"; allow(write) userdn=ldap:///self;)
```

- *Filtered roles.* The attributes that are part of the filter should be protected so that the user cannot relinquish the filtered role by modifying an attribute. The user should not be allowed to add, delete, or modify the attribute used by the filtered role. If the value of the filter attribute is computed, then all attributes that can modify the value of the filter attribute should be protected in the same way.
- *Nested roles.* A nested role is comprised of filtered and managed roles, so both ACIs should be considered for modifying the attributes (***nsRoleDN*** or something else) of the roles that comprise the nested role.

For more information about account inactivation, see [Section 20.16, “Manually Inactivating Users and Roles”](#).

## 8.3. AUTOMATICALLY CREATING DUAL ENTRIES

Some clients and integration with Red Hat Directory Server require dual entries. For example, both Posix systems typically have a group for each user. The Directory Server's *Managed Entries Plug-in* creates a new managed entry, with accurate and specific values for attributes, automatically whenever an appropriate origin entry is created.

### 8.3.1. About Managed Entries

The basic idea behind the Managed Entries Plug-in is that there are situations when Entry A is created and there should automatically be an Entry B with related attribute values. For example, when a Posix user (**posixAccount** entry) is created, a corresponding group entry (**posixGroup** entry) should also be created. An instance of the Managed Entries Plug-in identifies what entry (the *origin entry*) triggers the plug-in to automatically generate a new entry (the *managed entry*).

The plug-in works within a defined scope of the directory tree, so only entries within that subtree and that match the given search filter trigger a Managed Entries operation.

Much like configuring a class of service, a managed entry is configured through two entries:

- A definition entry, that identifies the scope of the plug-in instance and the template to use
- A template entry, that models what the final managed entry will look like

#### 8.3.1.1. About the Instance Definition Entry

As with the Linked Attributes and DNA Plug-ins, the Managed Entries Plug-in has a container entry in **cn=plugins,cn=config**, and each unique configuration instance of the plug-in has a definition entry beneath that container.

An instance of the Managed Entries Plug-in defines three things:

- The search criteria to identify the origin entries (using a search scope and a search filter)
- The subtree under which to create the managed entries (the new entry location)
- The template entry to use for the managed entries

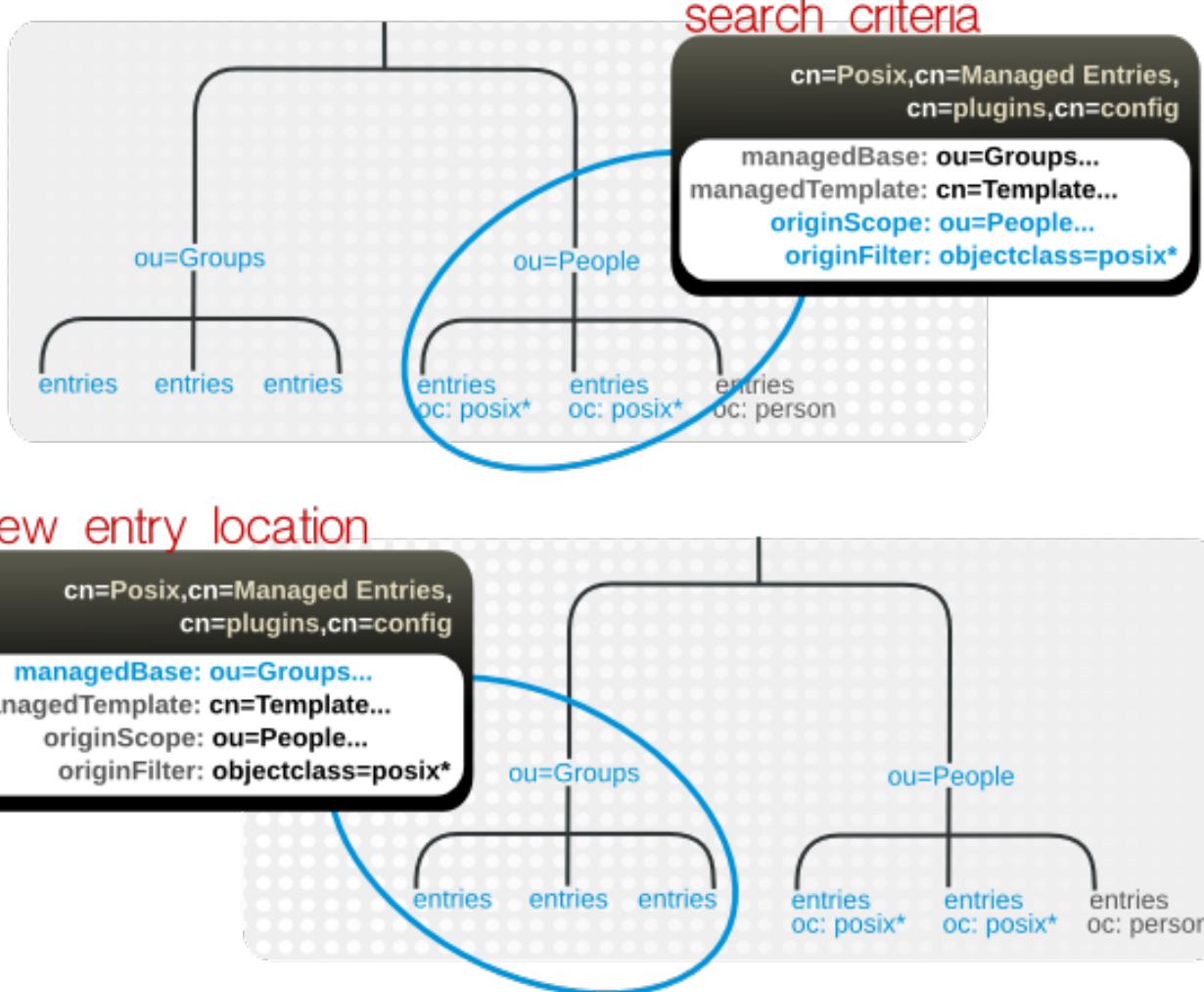


Figure 8.2. Defining Managed Entries

For example:

```

dn: cn=Posix User-Group,cn=Managed Entries,cn=plugins,cn=config
objectclass: extensibleObject
cn: Posix User-Group
originScope: ou=people,dc=example,dc=com
originFilter: objectclass=posixAccount
managedBase: ou=groups,dc=example,dc=com
managedTemplate: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
  
```

The origin entry does not have to have any special configuration or settings to create a managed entry; it simply has to be created within the scope of the plug-in and match the given search filter.

### 8.3.1.2. About the Template Entry

Each instance of the plug-in uses a template entry which defines the managed entry configuration. The template effectively lays out the entry, from the object classes to the entry values.



## NOTE

Since the template is referenced in the definition entry, it can be located anywhere in the directory. However, it is recommended that the template entry be under the replicated suffix so that any other suppliers in multi-supplier replication all use the same template for their local instances of the Managed Entries Plug-in.

The concept of a template entry is similar to the templates used in CoS, but there are some important differences. The managed entry template is slightly different than the type of template used for a class of service. For a class of service, the template contains a single attribute with a specific value that is fed into all of the entries which belong to that CoS. Any changes to the class of service are immediately reflected in the associated entries, because the CoS attributes in those entries are virtual attributes, not truly attributes set on the entry.

The template entry for the Managed Entries Plug-in, on the other hand, is not a central entry that supplies values to associated entries. It is a true template – it lays out what is in the entry. The template entry can contain both static attributes (ones with pre-defined values, similar to a CoS) and mapped attributes (attributes that pull their values or parts of values from the origin entry). The template is referenced when the managed entry is created and then any changes are applied to the managed entry *only* when the origin entry is changed and the template is evaluated again by the plug-in to apply those updates.

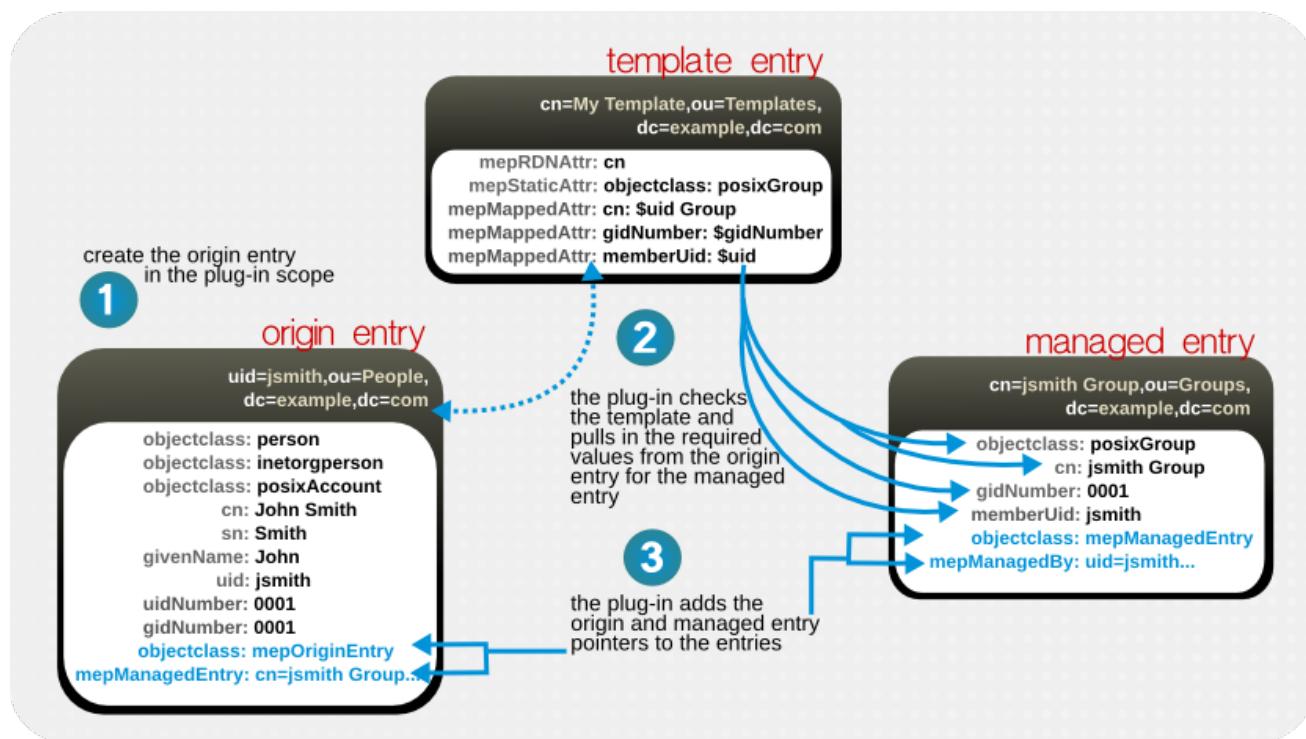


Figure 8.3. Templates, Managed Entries, and Origin Entries

The template can provide a specific value for an attribute in the managed entry by using a *static* attribute in the template. The template can also use a value that is derived from some attribute in the origin entry, so the value may be different from entry to entry; that is a *mapped* attribute, because it references the attribute type in the origin entry, not a value.

A mapped value uses a combination of token (dynamic values) and static values, but it can only use *one token in a mapped attribute*.

```
dn: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
objectclass: mepTemplateEntry
```

```

cn: Posix User-Group Template
mepRDNAtr: cn
mepStaticAttr: objectclass: posixGroup
mepMappedAttr: cn: $cn Group Entry
mepMappedAttr: gidNumber: $gidNumber
mepMappedAttr: memberUid: $uid

```

The mapped attributes in the template use tokens, prepended by a dollar sign (\$), to pull in values from the origin entry and use it in the managed entry. (If a dollar sign is actually in the managed attribute value, then the dollar sign can be escaped by using two dollar signs in a row.)

A mapped attribute definition can be quoted with curly braces, such as **Attr: \${cn}test**. Quoting a token value is not required if the token name is not immediately followed by a character that is valid in an attribute name, such as a space or comma. For example, **\$cn test** is acceptable in an attribute definition because a space character immediately follows the attribute name, but **\$cetest** is not valid because the Managed Entries Plug-in attempts to look for an attribute named **cetest** in the origin entry. Using curly braces identifies the attribute token name.



#### NOTE

Make sure that the values given for static and mapped attributes comply with the required attribute syntax.

#### 8.3.1.3. Entry Attributes Written by the Managed Entries Plug-in

Both the origin entry and the managed entry have special managed entries attributes which indicate that they are being managed by an instance of the Managed Entries Plug-in. For the origin entry, the plug-in adds links to associated managed entries.

```

dn: uid=jsmith,ou=people,dc=example,dc=com
objectclass: mepOriginEntry
objectclass: posixAccount
...
sn: Smith
mail: jsmith@example.com
mepManagedEntry: cn=jsmith Posix Group,ou=groups,dc=example,dc=com

```

On the managed entry, the plug-in adds attributes that point back to the origin entry, in addition to the attributes defined in the template.

```

dn: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
objectclass: mepManagedEntry
objectclass: posixGroup
...
mepManagedBy: uid=jsmith,ou=people,dc=example,dc=com

```

Using special attributes to indicate managed and origin entries makes it easy to identify the related entries and to assess changes made by the Managed Entries Plug-in.

#### 8.3.1.4. Managed Entries Plug-in and Directory Server Operations

The Managed Entries Plug-in has some impact on how the Directory Server carries out common operations, like add and delete operations.

**Table 8.3. Managed Entries Plug-in and Directory Server Operations**

Operation	Effect by the Managed Entries Plug-in
Add	<p>With every add operation, the server checks to see if the new entry is within the scope of any Managed Entries Plug-in instance. If it meets the criteria for an origin entry, then a managed entry is created and managed entry-related attributes are added to both the origin and managed entry.</p>
Modify	<p>If an origin entry is modified, it triggers the plug-in to update the managed entry. Changing a <i>template</i> entry, however, does not update the managed entry automatically. Any changes to the template entry are not reflected in the managed entry until after the next time the origin entry is modified.</p> <p>The mapped managed attributes <i>within</i> a managed entry cannot be modified manually, only by the Managed Entry Plug-in. Other attributes in the managed entry (including static attributes added by the Managed Entry Plug-in) can be modified manually.</p>
Delete	<p>If an origin entry is deleted, then the Managed Entries Plug-in will also delete any managed entry associated with that entry. There are some limits on what entries can be deleted.</p> <ul style="list-style-type: none"> <li>● A template entry cannot be deleted if it is currently referenced by a plug-in instance definition.</li> <li>● A managed entry cannot be deleted except by the Managed Entries Plug-in.</li> </ul>
Rename	<p>If an origin entry is renamed, then plug-in updates the corresponding managed entry. If the entry is moved <i>out</i> of the plug-in scope, then the managed entry is deleted, while if an entry is moved <i>into</i> the plug-in scope, it is treated like an add operation and a new managed entry is created. As with delete operations, there are limits on what entries can be renamed or moved.</p> <ul style="list-style-type: none"> <li>● A configuration definition entry cannot be moved out of the Managed Entries Plug-in container entry. If the entry is removed, that plug-in instance is inactivated.</li> <li>● If an entry is moved <i>into</i> the Managed Entries Plug-in container entry, then it is validated and treated as an active configuration definition.</li> <li>● A template entry cannot be renamed or moved if it is currently referenced by a plug-in instance definition.</li> <li>● A managed entry cannot be renamed or moved except by the Managed Entries Plug-in.</li> </ul>
Replication	<p>The Managed Entries Plug-in operations are <i>not initiated by replication updates</i>. If an add or modify operation for an entry in the plug-in scope is replicated to another replica, that operation does not trigger the Managed Entries Plug-in instance on the replica to create or update an entry. The only way for updates for managed entries to be replicated is to replicate the final managed entry over to the replica.</p>

### 8.3.2. Creating the Managed Entries Template Entry

The first entry to create is the template entry. The template entry must contain all of the configuration required for the generated, managed entry. This is done by setting the attribute-value assertions in static and mapped attributes in the template:

```
mepStaticAttr: attribute: specific_value
mepMappedAttr: attribute: $token_value
```

The static attributes set an explicit value; mapped attributes pull some value from the originating entry is used to supply the given attribute. The values of these attributes will be tokens in the form *attribute: \$attr*. As long as the syntax of the expanded token of the attribute does not violate the required attribute syntax, then other terms and strings can be used in the attribute. For example:

```
mepMappedAttr: cn: Managed Group for $cn
```

There are some syntax rules that must be followed for the managed entries:

- A mapped value use a combination of token (dynamic values) and static values, but it can only use *one token per mapped attribute*.
- The mapped attributes in the template use tokens, prepended by a dollar sign (\$), to pull in values from the origin entry and use it in the managed entry. (If a dollar sign is actually in the managed attribute value, then the dollar sign can be escaped by using two dollar signs in a row.)
- A mapped attribute definition can be quoted with curly braces, such as **Attr: \${cn}test**. Quoting a token value is not required if the token name is not immediately followed by a character that is valid in an attribute name, such as a space or comma. For example, **\$cn test** is acceptable in an attribute definition because a space character immediately follow the attribute name, but **\$cetest** is not valid because the Managed Entries Plug-in attempts to look for an attribute named **cetest** in the origin entry. Using curly braces identifies the attribute token name.
- Make sure that the values given for static and mapped attributes comply with the required attribute syntax.



#### NOTE

Make sure that the values given for static and mapped attributes comply with the required attribute syntax. For example, if one of the mapped attributes is **gidNumber**, then the mapped value should be an integer.

**Table 8.4. Attributes for the Managed Entry Template**

Attribute	Description
mepTemplateEntry (object class)	Identifies the entry as a template.
cn	Gives the common name of the entry.
mepMappedAttr	Contains an attribute-token pair that the plug-in uses to create an attribute in the managed entry with a value taken from the originating entry.

Attribute	Description
mepRDNAttr	Specifies which attribute to use as the naming attribute in the managed entry. The attribute used as the RDN <b>must</b> be a mapped attribute for the configuration to be valid.
mepStaticAttr	Contains an attribute-value pair that will be used, with that specified value, in the managed entry.

To create a template entry:

Use the **dsconf plugin managed-entries template add** command to add the template entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin managed-entries template
"cn=Posix User Template,ou=templates,dc=example,dc=com" add --rdn-attr "cn" --static-attr
"objectclass: posixGroup" --mapped-attr "cn: $cn Group Entry" "gidNumber: $gidNumber"
"memberUid: $uid"
```

### 8.3.3. Creating the Managed Entries Instance Definition

Once the template entry is created, then it is possible to create a definition entry that points to that template. The definition entry is an instance of the Managed Entries Plug-in.



#### NOTE

When the definition is created, the server checks to see if the specified template entry exists. If the template does not exist, then the server returns a warning that the definition configuration is invalid.

The definition entry must define the parameters to recognize the potential origin entry and the information to create the managed entry. The attributes available for the plug-in instance are listed in [Table 8.5, “Attributes for the Managed Entries Definition Entry”](#).

**Table 8.5. Attributes for the Managed Entries Definition Entry**

Attribute Name	Description
originFilter	The search filter to use to search for and identify the entries within the subtree which require a managed entry. The syntax is the same as a regular search filter.
originScope	The base subtree which contains the potential origin entries for the plug-in to monitor.
managedTemplate	Identifies the template entry to use to create the managed entry. This entry can be located anywhere in the directory tree.
managedBase	The subtree under which to create the managed entries.

**NOTE**

The Managed Entries Plug-in is enabled by default. If this plug-in is disabled, then re-enable it as described in [Section 1.10.3, “Enabling and Disabling Plug-ins”](#).

To create an instance:

1. Create the new plug-in instance below the **cn=Managed Entries,cn=plugins,cn=config** container entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin managed-entries config "cn=instance,cn=Managed Entries,cn=plugins,cn=config" add -- scope="ou=people,dc=example,dc=com" --filter="objectclass=posixAccount" --managed-base="ou=groups,dc=example,dc=com" --managed-template="cn=Posix User-Group Template,ou=Templates,dc=example,dc=com"
```

This command sets the scope and filter for the origin entry search, the location of the new managed entries, and the template entry to use.

2. If the Directory Server is not configured to enable dynamic plug-ins, restart the server to load the modified new plug-in instance:

```
# dsctl instance_name restart
```

### 8.3.4. Putting Managed Entries Plug-in Configuration in a Replicated Database

As [Section 8.3.1, “About Managed Entries”](#) highlights, different instances of the Managed Entries Plug-in are created as children beneath the container plug-in entry in **cn=plugins,cn=com**. (This is common for plug-ins which allow multiple instances.) The drawback to this is that the configuration entries in **cn=plugins,cn=com** are not replicated, so the configuration has to be re-created on each Directory Server instance.

The Managed Entries Plug-in entry allows the **nsslapd-pluginConfigArea** attribute. This attribute to another container entry, in the main database area, which contains the plug-in instance entries. This container entry can be in a replicated database, which allows the plug-in configuration to be replicated.

1. Create a container entry. For example, to create an entry that points back to the container entry, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin managed-entries set - -config-area="cn=managed entries container,ou=containers,dc=example,dc=com"
```

2. Move or create the definition ([Section 8.3.3, “Creating the Managed Entries Instance Definition”](#)) and template ([Section 8.3.2, “Creating the Managed Entries Template Entry”](#)) entries under the new container entry.

## 8.4. USING VIEWS

Virtual directory tree views, or views, create a virtual directory hierarchy, so it is easy to navigate entries, without having to make sure those entries physically exist in any particular place. The view uses information about the entries to place them in the view hierarchy, similarly to members of a filtered role or a dynamic group. Views superimpose a DIT hierarchy over a set of entries, and to client applications, views appear as ordinary container hierarchies.

### 8.4.1. About Views

Views create a directory tree similar to the regular hierarchy, such as using organizational unit entries for subtrees, but views entries have an additional object class (**nsview**) and a filter attribute (**nsviewfilter**) that set up a filter for the entries which belong in that view. Once the view container entry is added, all of the entries that match the view filter instantly populate the view. The target entries only appear to exist in the view; their true location never changes. For example, a view may be created as **ou=Location Views**, and a filter is set for **I=Mountain View**. Every entry, such as **cn=Jane Smith,I=Mountain View,ou=People,dc=example,dc=com**, is immediately listed under the **ou=Location Views** entry, but the real **cn=Jane Smith** entry remains in the **ou=People,dc=example,dc=com** subtree.

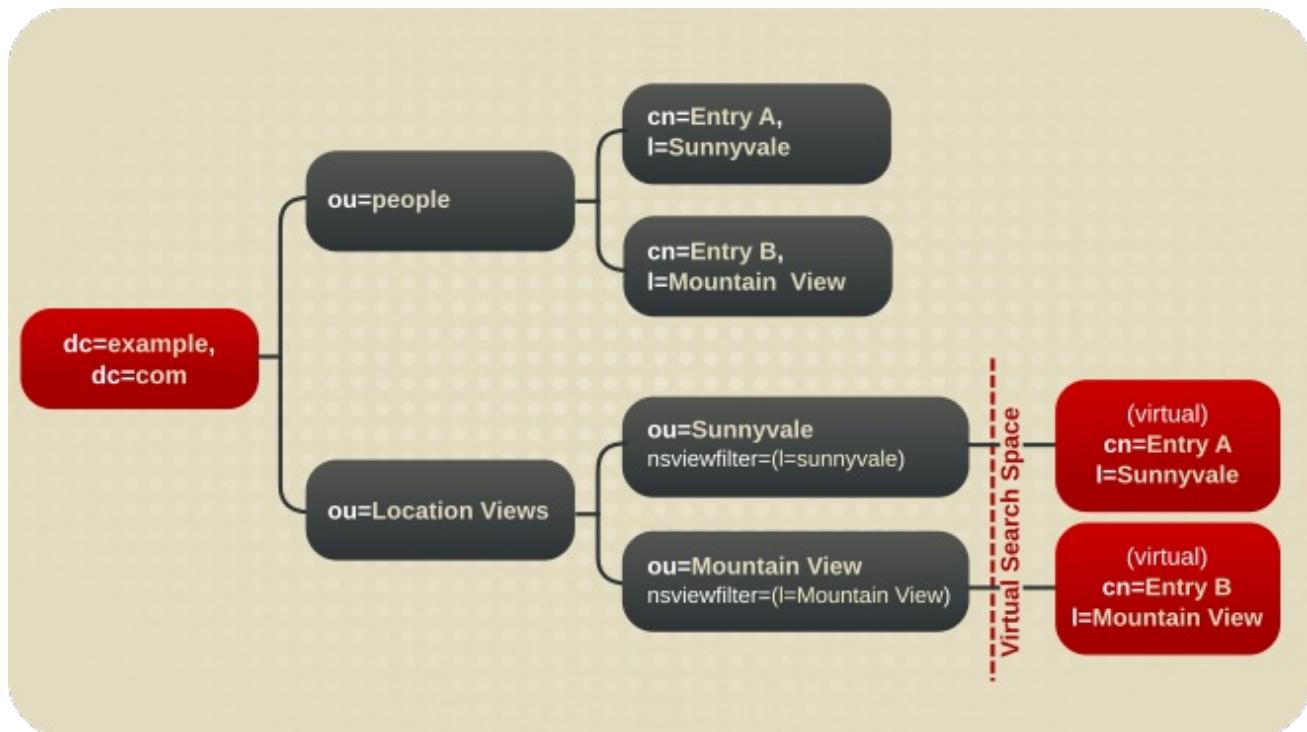


Figure 8.4. A Directory Tree with a Virtual DIT View hierarchy

Virtual DIT views behave like normal DITs in that a subtree or a one-level search can be performed with the expected results being returned.



#### NOTE

There is a sample LDIF file with example views entries, **Example-views.ldif**, installed with Directory Server. This file is in the **/usr/share/dirsrv/data/** directory. The sections in this chapter assume **Example-views.ldif** is imported to the server.

The *Red Hat Directory Server Deployment Guide* has more information on how to integrate views with the directory tree hierarchy.

### 8.4.2. Creating Views from the Command Line

1. Use the **ldapmodify** utility to bind to the server and prepare it to add the new view entry to the configuration file.
2. Assuming the view container **ou=Location Views,dc=example,dc=com** from the **Example-views.ldif** file is in the Directory Server, add the new views container entry, in this example, under the **dc=example,dc=com** root suffix. This entry must have the **nsview** object class and

the ***nsViewFilter*** attribute. The ***nsViewFilter*** attribute sets the attribute-value which identifies entries that belong in the view.

```
dn: ou=Mountain View,ou=Location Views,dc=example,dc=com
changetype: add
objectClass: top
objectClass: organizationalUnit
objectClass: nsview
ou: Mountain View
nsViewFilter: l=Mountain View
description: views categorized by location
```

### 8.4.3. Improving Views Performance

As [Section 8.4.1, "About Views"](#) describes, views are derived from search results based on a given filter. Part of the filter is the attribute defined in the ***nsViewFilter*** attribute; the rest of the filter is based on the entry hierarchy, looking for the ***entryid*** and ***parentid*** of the real entries included in the view.

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

If any of the searched-for attributes – ***entryid***, ***parentid***, or the attribute set in ***nsViewFilter*** – are not indexed, then the views search becomes an unindexed search because the views operation searches the entire tree for matching entries.

To improve views performance, create equality indexes for ***entryid***, ***parentid***, and the attribute set in ***nsViewFilter***.

Creating equality indexes is covered in [Section 13.2, "Creating Standard Indexes"](#), and updating existing indexes to include new attributes is covered in [Section 13.3, "Creating New Indexes to Existing Databases"](#).

## 8.5. MANAGING ORGANIZATIONAL UNITS

Administrators can use organizational units (OU) as a container for directory entries. For example, you can use OUs to separate user and group entries. To manage OUs in Directory Server, use the **dsidm organizationalunit** command.

- To create an OU, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
organizationalunit create --ou OU_name
```

- To list the OUs in an entry, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
organizationalunit list
People
...
```

- To rename an OU, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
organizationalunit rename old_name new_name
```

- To delete an OU, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"  
organizationalunit delete OU_name
```

# CHAPTER 9. CONFIGURING SECURE CONNECTIONS

By default, clients and users connect to the Red Hat Directory Server over a standard connection. Standard connections do not use any encryption, so information is sent back and forth between the server and client in the clear.

Directory Server supports TLS connections, **STARTTLS** connection, and SASL authentication, which provide layers of encryption and security that protect directory data from being read even if it is intercepted.

## 9.1. REQUIRING SECURE CONNECTIONS

Directory Server provides the following ways of using encrypted connections:

### LDAPS

When you use the LDAPS protocol, the connection starts using encryption and either succeeds or fails. However, no unencrypted data is ever send over the network. For this reason, prefer LDAPS instead of using **STARTTLS** over unencrypted LDAP.

### STARTTLS over LDAP

Clients establish an unencrypted connection over the LDAP protocol and then send the **STARTTLS** command. If the command succeeds, all further communication is encrypted.



#### WARNING

If the **STARTTLS** command fails and the client does not cancel the connection, all further data, including authentication information, is sent unencrypted over the network.

### SASL

Simple Authentication and Security Layer (SASL) enables you to authenticate a user using external authentication methods, such as Kerberos. For details, see [Section 9.10, “Setting up SASL Identity Mapping”](#).

## 9.2. SETTING A MINIMUM STRENGTH FACTOR

For additional security, the Directory Server can be configured to require a certain level of encryption before it allows a connection. The Directory Server can define and require a specific Security Strength Factor (SSF) for any connection. The SSF sets a minimum encryption level, defined by its key strength, for any connection or operation.

To require a minimum SSF for any and all directory operations, set the **nsslapd-minssf** configuration attribute. When enforcing a minimum SSF, Directory Server looks at each available encryption type for an operation – TLS or SASL – and determines which has the higher SSF value and then compares the higher value to the minimum SSF. It is possible for both SASL authentication and TLS to be configured for some server-to-server connections, such as replication.

**NOTE**

Alternatively, use the ***nsslapd-minssf-exclude-rootdse*** configuration attribute. This sets a minimum SSF setting for all connections to the Directory Server except for queries against the root DSE. A client may need to obtain information about the server configuration, like its default naming context, before initiating an operation. The ***nsslapd-minssf-exclude-rootdse*** attribute allows the client to get that information without having to establish a secure connection first.

The SSF for a connection is evaluated when the first operation is initiated on a connection. This allows **STARTTLS** and SASL binds to succeed, even though those two connections initially open a regular connection. After the TLS or SASL session is opened, then the SSF is evaluated. Any connection which does not meet the SSF requirements is closed with an LDAP unwilling to perform error.

Set a minimum SSF to disable insecure connections to a directory.

**WARNING**

If you connect to the directory using the unencrypted LDAP protocol without SASL, the first LDAP message can contain the bind request. In this case, the credentials are sent unencrypted over the network before the server cancels the connection, because the SSF did not meet the minimum value set.

Use the LDAPS protocol or SASL binds to ensure that the credentials are never sent unencrypted.

The default ***nsslapd-minssf*** attribute value is 0, which means there is no minimum SSF for server connections. The value can be set to any reasonable positive integer. The value represents the required key strength for any secure connection.

The following example sets the ***nsslapd-minssf*** parameter to **128**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-minssf=128
Successfully replaced "nsslapd-minssf"
```

**NOTE**

An ACI can be set to require an SSF for a specific type of operation, as in [Section 18.10.2.4, “Requiring a Certain Level of Security in Connections”](#).

Secure connections can be required for bind operations by turning on the ***nsslapd-require-secure-binds*** attribute, as in [Section 20.12.1, “Requiring Secure Binds”](#).

## 9.3. MANAGING THE NSS DATABASE USED BY DIRECTORY SERVER

To use TLS encryption or certificate-based authentication, you must manage the certificates in a Network Security Services (NSS) database. When you created the instance, the **dscreate** utility automatically created this database in the **/etc/dirsrv/slapd-instance\_name** directory and protected it with a strong password. The utility stored the password in the

`/etc/dirsrv/slappd-instance_name/pwdfile.txt` file. Note that Directory Server does not use this file. The **dscreate** utility only created this file to provide the password to the administrator. For details about changing the password, see [Section 9.3.10, “Changing the Password of the NSS Database”](#).

This section describes the most frequent actions about managing the Directory Server's NSS database.

### 9.3.1. Creating a Certificate Signing Request

The Certificate Signing Request (CSR) is a request to the Certificate Authority (CA) to sign the key of the server. This section describes how to create the CSR including the private key.



#### NOTE

Directory Server supports only creating a private key and CSR directly in the NSS database using the **certutil** utility.

#### 9.3.1.1. Creating a Certificate Signing Request Using the Command Line

To create the key and a CSR, use the **dsctl tls generate-server-cert-csr** command:

```
# dsctl instance_name tls generate-server-cert-csr -s "certificate_subject"
```

The **dsctl tls generate-server-cert-csr** command stores the CSR in the `/etc/dirsrv/slappd-instance_name/Server-Cert.csr` file and the private key in the Directory Server's network security services (NSS) database.

#### Example 9.1. Creating a Private Key and CSR for a Single Host Name

The following command generates a bit private key for the **server.example.com** host:

```
# dsctl instance_name tls generate-server-cert-csr -s  
"CN=server.example.com,O=example_organization,OU=IT,ST=North Carolina,C=US"
```

The string specified in the **-s** parameter must be a valid subject name according to [RFC 1485](#). The **CN** field is required, and you must set it to the Fully-qualified Domain Name (FQDN) of the server. The other fields are optional.

#### Example 9.2. Creating a Private Key and CSR for a Multi-homed Host

If a Directory Server host has multiple names, create a CSR with all host names in the SAN extension of the CSR. The following command generates a bit private key and a CSR for the **server.example.com** and **server.example.net** host names:

```
# dsctl instance_name tls generate-server-cert-csr -s  
"CN=server.example.com,O=example_organization,OU=IT,ST=North Carolina,C=US"  
server.example.com server.example.net
```

If you specify the host names as the last parameters, the command adds the SAN extension with the **DNS:server.example.com**, **DNS:server.example.net** entries to the CSR. The string specified in the **-s** parameter must be a valid subject name according to [RFC 1485](#). The **CN** field is required, and you must set it to one of the FQDNs of the server. The other fields are optional.

After you have generated the CSR, submit it to the CA to get a certificate issued. For further details, see your CA's documentation.

### 9.3.2. Installing a CA Certificate

To enable Directory Server to trust the Certificate Authority (CA) you must install the certificate of the CA into the Network Security Services (NSS) database. During this process, you must set which certificates issued by the CA should be trusted:

**Table 9.1. CA Trust Options**

Web Console Option	dsconf and certutil Option	Description
<b>(C) Trusted CA</b>	<b>C,,</b>	The server verifies that certificates, used to establish an encrypted connection to a replication partner, have been issued by a trusted CA.
<b>(T) Trusted CA Client Auth</b>	<b>T,,</b>	The server trusts this CA certificate for issuing client certificates suitable for TLS <b>EXTERNAL</b> binds.

You can set both options for a CA. When you use **certutil**, pass the **-T "CT,,"** parameter to the utility.

#### 9.3.2.1. Installing a CA Certificate Using the Command Line

To install a CA certificate:

1. Import the CA certificate. For example, to import the CA certificate stored in the **/root/ca.crt** file and store it in the database with the **Example CA** nick name:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate add --file /root/ca.crt --name "Example CA"
```

2. Set the trust options. For example, to set the **CT,,** trust flags:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate set-trust-flags "Example CA" --flags "CT,,"
```

#### 9.3.2.2. Installing a CA Certificate Using the Web Console

To install a CA certificate using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the **Security** entry.
4. Open the **Certificate Management** tab, and select the **Trusted Certificate Authorities** sub-tab.

5. Click **Add CA Certificate**.
6. Enter the path to the CA certificate file and a nickname for the certificate.

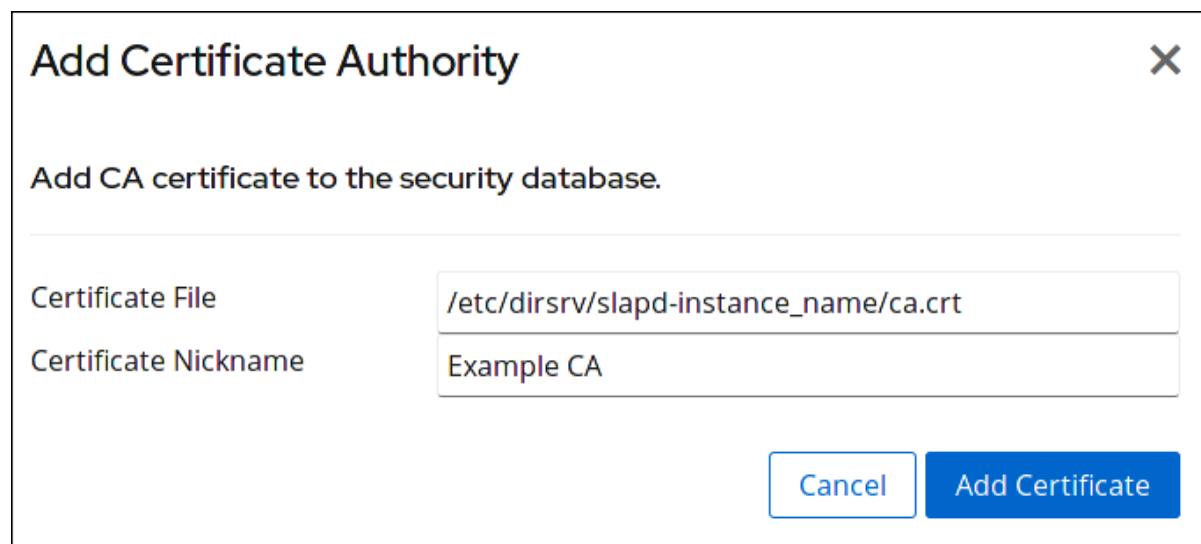


Figure 9.1. Adding a CA Certificate



**NOTE**

The CA certificate must be stored locally on the Directory Server host and must be readable by the **dirsrv** user.

7. Click **Add Certificate**.
8. Click **Actions** next to the imported CA certificate, and select **Edit Trust Flags**.
9. Select **(C) - Trusted CA** and **(T) - Trusted CA Client Auth** in the **SSL** column.

Flags	SSL	Email	Object Signing
(C) - Trusted CA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(T) - Trusted CA Client Auth	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(c) - Valid CA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(P) - Trusted Peer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(p) - Valid Peer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(u) - Private Key	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cancel
Save

Figure 9.2. Adding Trust Flags of a CA Certificate

### 9.3.3. Importing a Private Key and Server Certificate

This section describes how to import both a private key and Certificate Signing Request (CSR), if you did not create them in the NSS database using an external tool.

If you created the private key and CSR in the NSS database, follow the procedure described in [Section 9.3.4, “Installing a Server Certificate”](#).

To import the certificate from the **/root/server.crt** and the private key from the **/root/server.key** file, enter:

```
# dsctl instance_name tls import-server-key-cert /root/server.crt /root/server.key
```

Note that the **dsctl tls import-server-key-cert** command requires the paths in the following order:

1. Path to the server certificate.
2. Path to the private key file.

### 9.3.4. Installing a Server Certificate

After the Certificate Authority (CA) issued the requested certificate, you must install it in the Network Security Services (NSS) database.

If you created the private key and certificate signing request not in the NSS database, follow the procedure described in [Section 9.3.3, “Importing a Private Key and Server Certificate”](#)

#### 9.3.4.1. Installing a Server Certificate Using the Command Line

To install a server certificate in the Directory Server's NSS database, use the **certutil** utility. For example:

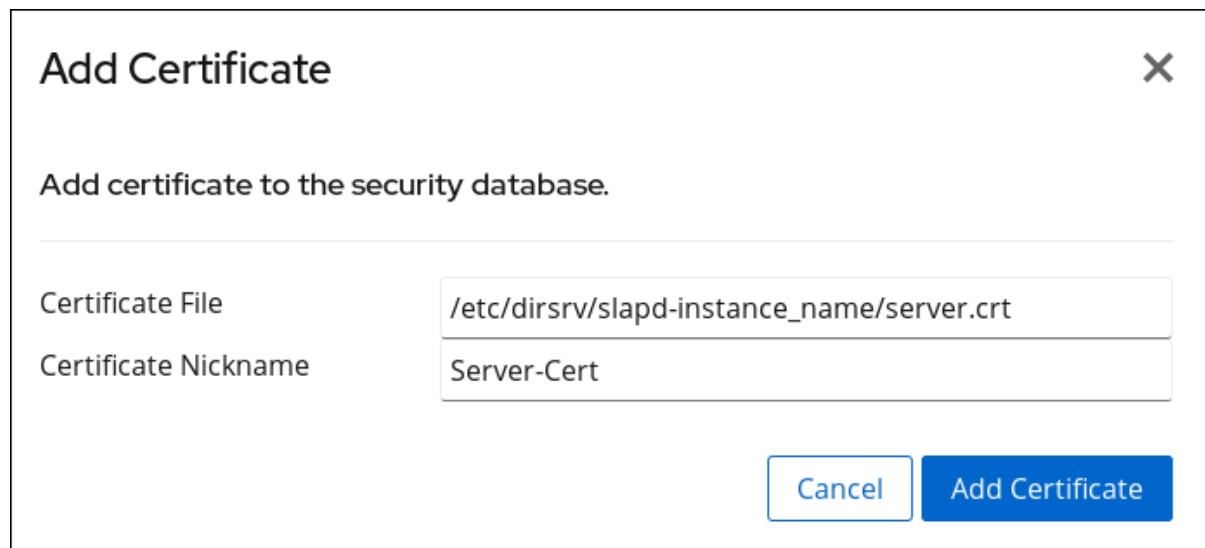
1. Install the CA certificate. See [Section 9.3.2, "Installing a CA Certificate"](#).
2. Import the server certificate. For example to import the certificate stored in the **/root/instance\_name.crt** file, and set it as the primary certificate the instance uses:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security certificate add --file /root/instance_name.crt --name "Server-Cert" --primary-cert
```

#### 9.3.4.2. Installing a Server Certificate Using the Web Console

To install a server certificate using the web console:

1. Install the CA certificate. See [Section 9.3.2, "Installing a CA Certificate"](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
3. Select the instance.
4. Open the **Server Settings** menu, and select the **Security** entry.
5. Open the **Certificate Management** tab, and select the **TLS Certificates** sub-tab.
6. Click **Add Server Certificate**.
7. Enter the path to the server certificate file and a nickname for the certificate.



**Figure 9.3. Adding a Server Certificate**



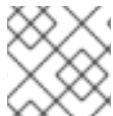
#### NOTE

The server certificate must be stored locally on the Directory Server host and must be readable by the **dirsrv** user.

8. Click **Add Certificate**.

### 9.3.5. Generating and Installing a Self-signed Certificate

When you created the instance with TLS enabled using the **dscreate** utility, **dscreate** automatically created and installed a self-signed certificate. However, if you did not enable TLS during instance creation, you can manually create and install a self-signed certificate.



#### NOTE

You can only perform this operation using the command line.

To create and install a self-signed certificate:

1. Generate a noise file with random data. For example, to generate a file with a size of 4096 bits:

```
# openssl rand -out /tmp/noise.bin 4096
```

2. Create the self-signed certificate and add it to the NSS database:

```
# certutil -S -x -d /etc/dirsrv/slapd-instance_name -z /tmp/noise.bin \
-n "Server-Cert" -s "CN=$HOSTNAME" -t "CT,C,C" -m $RANDOM \
--keyUsage digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment
```

Red Hat Enterprise Linux automatically replaces the **\$HOSTNAME** variable with the Fully Qualified Domain Name (FQDN) and **\$RANDOM** with a randomly-generated number. For further details about the parameters used in the previous commands, see the `certutil(1)` man page.

3. Optionally, verify that the generated certificate is self-signed:

```
# certutil -L -d /etc/dirsrv/slapd-instance_name -n "Server-Cert" | egrep "Issuer|Subject"
Issuer: "CN=server.example.com"
Subject: "CN=server.example.com"
```

The output of this command must display the FQDN of the Directory Server host for both the issuer and subject of the certificate.

### 9.3.6. Renewing a Certificate

If a certificate will expire in the near future, you must renew it in time to continue establishing secure connections.

#### 9.3.6.1. Renewing a Certificate Using the Command Line

To renew the server certificate:

- If you do not use attribute encryption:

1. Create a new Certificate Signing Request (CSR) with the same options, such as key size, host name, and subject. For details about creating a CSR, see [Section 9.3.1.1, “Creating a Certificate Signing Request Using the Command Line”](#)
2. After you received the issued certificate from your CA, install it in the database using the same nickname. See [Section 9.3.2.1, “Installing a CA Certificate Using the Command Line”](#).

Directory Server will automatically use the newer issued certificate.

- If you use attribute encryption, see [Section 10.5, “Updating the TLS Certificates Used for Attribute Encryption”](#).

### 9.3.7. Removing a Certificate

If a certificate is no longer needed, for example, because it has been exposed, remove it from the database.

#### 9.3.7.1. Removing a Certificate Using the Command Line

To remove a certificate using the command line:

1. Optionally, display the certificates in the database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security certificate list  
  
Certificate Name: Server-Cert  
Subject DN: CN=server.example.com  
Issuer DN: CN=Example CA  
Expires: 2022-07-29 11:10:14  
Trust Flags: ,,
```

2. Remove the certificate. For example, to remove the certificate with the Server-Cert nickname:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security certificate del  
Server-Cert
```

#### 9.3.7.2. Removing a Certificate Using the Web Console

To remove a certificate using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the **Security** entry.
4. Open the **Certificate Management** tab, and select the **TLS Certificates** sub-tab.
5. Click **Actions** next to the certificate, and select **Delete Certificate**.
6. Click **Yes**.

### 9.3.8. Removing a Private Key

If a private key is no longer needed, for example, because you created a stronger key, remove it from the database.

**WARNING**

If you remove a private key, certificates based on this key are no longer working.

### 9.3.8.1. Removing a Private Key Using the Command Line

To remove a private key:

1. Remove all certificates based on the key you want to delete. See [Section 9.3.7, "Removing a Certificate"](#).
2. Optionally, display the keys in the database:

```
# certutil -d /etc/dirsrv/slapd-instance_name -K
certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key and Certificate Services"
Enter Password or Pin for "NSS Certificate DB":
< 0> rsa    7a2fb6c269d83c4036eac7e4edb6aaaf2ed08bc4a  Server-Cert
< 1> rsa    662b826aa3dd4ca7fd7e6883558cf3866c42f4e2  example-cert
```

3. Remove the private key. For example, to remove the private key with the *example-cert* nickname:

```
# certutil -d /etc/dirsrv/slapd-instance_name -F -n "example-cert"
```

### 9.3.9. Changing the CA Trust Options

In certain situations you need to update the trust option of a Certificate Authority (CA). This section describes this procedure.

#### 9.3.9.1. Changing the CA Trust Options Using the Command Line

To change the trust options of a CA, pass the new options in the **--flags** parameter to the **dsconf security ca-certificate set-trust-flags** command.

For example, to set that Directory Server trusts only client authentication certificates issued by the CA named **example-CA**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate set-trust-flags
"example-CA" --flags "T,,"
```

The **--flags trust\_options** parameter sets which certificates issued by the CA should be trusted. See [Table 9.1, "CA Trust Options"](#).

#### 9.3.9.2. Changing the CA Trust Options Using the Web Console

To change the trust options of a CA using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the **Security** entry.
4. Open the **Certificate Management** tab.
5. On the **Trusted Certificate Authorities** sub-tab, click **Actions** next to the imported CA certificate, and select **Edit Trust Flags**.
6. Select the trust flags. For example:

**Edit Certificate Trust Flags**

Flags	SSL	Email	Object Signing
(C) - Trusted CA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(T) - Trusted CA Client Auth	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(c) - Valid CA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(P) - Trusted Peer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(p) - Valid Peer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
(u) - Private Key	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cancel Save

**Figure 9.4. Setting the Trust Flags of a CA Certificate**

7. Click **Save**.

### 9.3.10. Changing the Password of the NSS Database

In certain situations, administrators want to change the password of the Network Security Services (NSS) database. This section describes this process.



#### IMPORTANT

If you use a password file to enable Directory Server to automatically open the Network Security Services (NSS) database, you must update the file after you set the new password. See [Section 9.4.1.5, "Creating a Password File for Directory Server"](#).

#### 9.3.10.1. Changing the Password of the NSS Database Using the Command Line

To change the password of the NSS database:

```
# certutil -d /etc/dirsrv/slapd-instance_name -W
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.
```

```
Enter new password:
Re-enter password:
Password changed successfully.
```

## 9.4. ENABLING TLS

Directory Server supports encrypted connections between clients and the server, as well as between servers in a replication environment. For this, Directory Server supports:

- The LDAPS protocol: TLS encryption is used directly after the connection has been established.
- The **STARTTLS** command over the LDAP protocol: The connection is unencrypted until the client sends the **STARTTLS** command.



### IMPORTANT

For security reasons, Red Hat recommends enabling TLS encryption.

You can use TLS with simple authentication using a bind Distinguished Name (DN) and password, or using certificate-based authentication.

Directory Server's cryptographic services are provided by Mozilla Network Security Services (NSS), a library of TLS and base cryptographic functions. NSS includes a software-based cryptographic token which is Federal Information Processing Standard (FIPS) 140-2 certified.

### 9.4.1. Enabling TLS in Directory Server

This section describes how to enable TLS in Directory Server.

#### 9.4.1.1. Enabling TLS in Directory Server Using the Command Line

To enable TLS using the command line:

1. Request and install the certificate:
  - For a certificate issued by a Certificate Authority (CA):
    1. Create a Certificate Signing Request (CSR). See [Section 9.3.1.1, “Creating a Certificate Signing Request Using the Command Line”](#)
    2. Import the CA certificate. See [Section 9.3.2.1, “Installing a CA Certificate Using the Command Line”](#).
    3. Import the server certificate issued by the CA. See [Section 9.3.4.1, “Installing a Server Certificate Using the Command Line”](#).

- For a self-signed certificate, see [Section 9.3.5, “Generating and Installing a Self-signed Certificate”](#).
2. Enable TLS and set the LDAPS port:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-securePort=636 nsslapd-security=on
Successfully replaced "nsslapd-securePort"
Successfully replaced "nsslapd-security"
```

3. Display the name of the server certificate in the NSS database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security certificate list

Certificate Name: Server-Cert
Subject DN: CN=server.example.com
Issuer DN: CN=Example CA
Expires: 2022-07-29 11:10:14
Trust Flags: ,,
```

You need the nickname in the next step.

4. To enable the RSA cipher family, setting the NSS database security device, and the server certificate name:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security rsa set --tls-allow-rsa-certificates on --nss-token "internal (software)" --nss-cert-name Server-Cert
```



#### NOTE

By default, the name of the security device in the NSS database is **internal (software)**.

5. Optionally, update the list of ciphers Directory Server supports. For details, see [Section 9.4.1.3.2, “Displaying and Setting the Ciphers Used by Directory Server Using the Command Line”](#).
6. Optionally, enable certificate-based authentication. For details, see [Section 9.9, “Using Certificate-based Client Authentication”](#).
7. Optionally, create a password file to enable Directory Server to start without prompting for the password of the NSS database. For details, see [Section 9.4.1.5, “Creating a Password File for Directory Server”](#).
8. Restart the Directory Server instance:

```
# dsctl instance_name restart
```

If you set a password on the NSS database and did not create a password file, Directory Server prompts for the password of the NSS database. For details, see [Section 9.4.1.4, “Starting Directory Server Without a Password File”](#).

#### 9.4.1.2. Enabling TLS in Directory Server Using the Web Console

To enable TLS in Directory Server using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Create a CSR. See [Section 9.3.1, “Creating a Certificate Signing Request”](#).
4. Import the Certificate Authority (CA) certificate. See [Section 9.3.2.2, “Installing a CA Certificate Using the Web Console”](#).
5. Import the server certificate issued by the CA. See [Section 9.3.4.2, “Installing a Server Certificate Using the Web Console”](#).
6. Open the **Server Settings** menu, and select the **Security** entry.
7. On the **Security Configuration** tab:
  - a. Click **Security Enabled**.
  - b. Select the certificate's nickname in the **Server Certificate Name** field.
  - c. Optionally, change the settings for the minimum and maximum TLS version that the server should support.
  - d. Optionally, configure client authentication to enable users to authenticate using certificates. For details, see [Section 9.9, “Using Certificate-based Client Authentication”](#).
8. Click **Save Configuration**.
9. Optionally, create a password file to enable Directory Server to start without prompting for the password of the NSS database. For details, see [Section 9.4.1.5, “Creating a Password File for Directory Server”](#).
10. Restart the Directory Server instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#)

If you set a password on the NSS database and did not create a password file, Directory Server prompts for the password of the NSS database. For details, see [Section 9.4.1.4, “Starting Directory Server Without a Password File”](#).

### 9.4.1.3. Setting Encryption Ciphers

Directory Server supports different ciphers, and you can enable or disable them. A cipher is the algorithm used in encryption. When a client initiates a TLS connection with a server, the client tells the server what ciphers it prefers to encrypt information. If the server supports at least one of these ciphers, the encrypted connection can be established using this algorithm.

If you enabled encryption according to [Section 9.4, “Enabling TLS”](#), you can display and update the ciphers Directory Server uses.

#### 9.4.1.3.1. Displaying the Default Ciphers

If the **nsSSL3Ciphers** parameter is not set in the **cn=encryption,cn=config** entry, Directory Server uses the default ciphers of the Network Security Service (NSS). To display the default ciphers:



```
# /usr/lib64/nss/unsupported-tools/listsuites | grep -B1 --no-group-separator "Enabled"
TLS_AES_128_GCM_SHA256:
 0x1301 TLS 1.3 TLS 1.3 AES-GCM 128 AEAD Enabled FIPS Domestic
TLS_CHACHA20_POLY1305_SHA256:
 0x1303 TLS 1.3 TLS 1.3 CHACHA20POLY1305 256 AEAD Enabled Domestic
...
```

#### 9.4.1.3.2. Displaying and Setting the Ciphers Used by Directory Server Using the Command Line

##### Displaying all Available Ciphers

To display the list of all available ciphers supported in Directory Server:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ciphers list --supported
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
...
```

This is only a list of available ciphers you can enable or disable. The list does not display the ciphers Directory Server currently uses.

##### Displaying the Ciphers Directory Server Uses

To display the ciphers Directory Server currently uses, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ciphers list --enabled
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
...
```

Additionally, you can display the ciphers which are configured to be enabled and disabled:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ciphers list
default
+tls_rsa_aes_128_sha
+tls_rsa_aes_256_sha
...
```

The **default** keyword refers to the preferred default ciphers provided by the NSS. See [Section 9.4.1.3.1, “Displaying the Default Ciphers”](#).



#### IMPORTANT

Directory Server uses the settings from the ***nsSSL3Ciphers*** attribute to generate the list of ciphers which are actually used. However, if you enabled weak ciphers in ***nsSSL3Ciphers***, but set the ***allowWeakCiphers*** parameter to **off**, which is the default, Directory Server only uses the strong ciphers and displays them in the ***nsSSLSupportedCiphers*** read-only attribute.

#### Updating the List of Enabled Ciphers

To update the list of enabled ciphers:

1. Display the list of currently enabled ciphers. See [the section called “Displaying the Ciphers Directory Server Uses”](#).

- To enable only specific ciphers, update the ***nsSSL3Ciphers*** attribute. For example, to enable only the **TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256** cipher:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ciphers set "-all,+TLS_RSA_WITH_AES_128_GCM_SHA256"
```

- Restart the Directory Server instance:

```
# dsctl instance_name restart
```

- Optionally, display the list of enabled ciphers to verify the result. See [the section called “Displaying the Ciphers Directory Server Uses”](#).

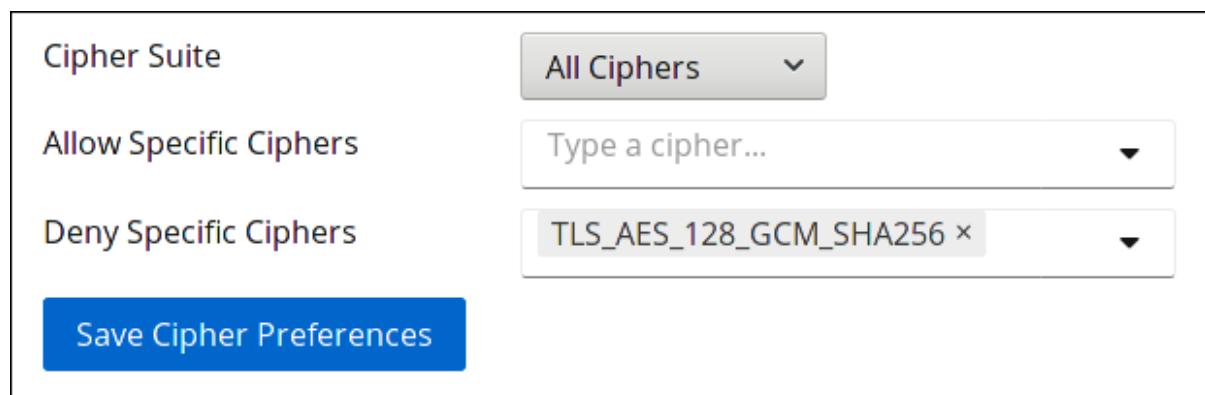
#### 9.4.1.3.3. Displaying and Setting the Ciphers Used by Directory Server Using the Web Console

To select and optionally update the ciphers using the web console:

- Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
- Select the instance.
- Open the **Server Settings** menu, and select the **Security** entry.
- On the **Cipher Preferences** tab, Directory Server displays the currently enabled ciphers.
- If you use different ciphers than the default, select **Default Ciphers** in the **Ciphers Suite** field to automatically enable the default ciphers. For details, see [Section 9.4.1.3.1, “Displaying the Default Ciphers”](#).

Alternatively, you can set **Ciphers Suite** to:

- All Ciphers** to enable all ciphers. Optionally, disable specific ciphers in the **Deny Specific Ciphers** field.
- No Ciphers** to disable all ciphers. Optionally, enable specific ciphers in the **Allow Specific Ciphers** field.



- Click **Save Cipher Preferences**.
- If you updated the list of ciphers, restart the Directory Server instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#)

#### 9.4.1.4. Starting Directory Server Without a Password File

If you start Directory Server with encryption enabled and a password set on the NSS database:

- If the **ns-slapd** Directory Server process is started by the **systemctl** command, **systemd** prompts for the password and automatically passes the input to the **systemd-tty-ask-password-agent** utility. For example:

```
# systemctl start dirsrv@instance_name
Enter PIN for Internal (Software) Token:
```

- In rare cases, when the **ns-slapd** Directory Server process is not started by the **systemctl** utility and is detached from the terminal, a message is sent to all terminals using the **wall** command. For example:

```
Broadcast message from root@server (Fri 2017-01-01 06:00:00 CET):
```

```
Password entry required for 'Enter PIN for Internal (Software) Token:' (PID 1234).
Please enter password with the systemd-tty-ask-password-agent tool!
```

To enter the password, run:

```
# systemd-tty-ask-password-agent
Enter PIN for Internal (Software) Token:
```

#### 9.4.1.5. Creating a Password File for Directory Server

If encryption is enabled and a password set on the NSS database, Directory Server prompts for this password when the service starts. See [Section 9.4.1.4, "Starting Directory Server Without a Password File"](#).

To bypass this prompt, you can store the NSS database password in the **/etc/dirsrv/slappd-*instance\_name*/pin.txt** file. This enables Directory Server to start automatically without prompting for this password.



#### WARNING

The password is stored in clear text. Do not use a password file if the server is running in an unsecured environment.

To create the password file:

1. Create the **/etc/dirsrv/slappd-*instance\_name*/pin.txt** file with the following content:

- If you use the NSS software cryptography module, which is the default:

```
Internal (Software) Token:password
```

- If you use a Hardware Security Module (HSM):

*name\_of\_the\_token:password*

2. Set the permissions:

```
# chown dirsrv:dirsrv /etc/dirsrv/slappd-instance_name/pin.txt
# chmod 400 /etc/dirsrv/slappd-instance_name/pin.txt
```

#### 9.4.1.6. Managing How Directory Server Behaves If the Certificate Has Been Expired

By default, if encryption is enabled and the certificate has expired, Directory Server logs a warning and the service starts. To change this behavior, set the **nsslapd-validate-cert** parameter. You can set it to the following values:

- **warn**: The Directory Server instance starts and log a warning about the expired certificate into the **/var/log/dirsrv/slappd-*instance\_name*/error** log file. This is the default setting.
- **on**: Directory Server validates the certificate and the instance fails to start if the certificate has expired.
- **off**: Directory Server does not validate the certificate expiration date. The instance starts and no warning will be logged.

#### Example 9.3. Preventing Directory Server to Start If the Certificate Has Been Expired

To prevent Directory Server from starting if the certificate has expired:

1. Set the **nsslapd-validate-cert** parameter to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
validate-cert=on
Successfully replaced "nsslapd-validate-cert"
```

2. Restart the Directory Server instance:

```
# dsctl instance_name restart
```

#### 9.4.2. Adding the CA Certificate Used By Directory Server to the Trust Store of Red Hat Enterprise Linux

When you enabled TLS encryption in Directory Server, you configured the instance to use a certificate issued by a CA. If a client now establishes a connection to the server using the LDAPS protocol or the **STARTTLS** command over LDAP, Directory Server uses this certificate to encrypt the connection. Client utilities use the CA certificate to verify if the server's certificate is valid. By default, these utilities cancel the connection if they do not trust the certificate of the server.

#### Example 9.4. Possible Connection Errors If Client Utilities Do Not Use the CA Certificate

If client utilities do not use the CA certificate, the utilities cannot validate the server's certificate when using TLS encryption. As a consequence, the connection to the server fails. For example:

- **dsconf**

```
# dsconf -D "cn=Directory Manager" ldaps://server.example.com:636 config get
Error: {'desc': "Can't contact LDAP server", 'info': 'error:1416F086:SSL
routines:tls_process_server_certificate:certificate verify failed (self signed certificate in
certificate chain)'}
```

- **ldapsearch**

```
# ldapsearch -H ldaps://server.example.com:636 -D "cn=Directory Manager" -W -b
"dc=example,dc=com" -x
Enter LDAP Password:
ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```

To enable client utilities on Red Hat Enterprise Linux to verify the certificate that Directory Server uses, add the CA certificate to the trust store of the operating system:

1. If you do not have a local copy of the CA certificate used by Directory Server:

- a. List the certificates in the server's NSS database:

# certutil -d /etc/dirsrv/slapd-instance_name/-L	
Certificate Nickname	Trust Attributes
	SSL,S/MIME,JAR/XPI
<i>Example CA</i>	C,,
Server-Cert	u,u,u

- b. Use the nickname of the CA certificate in the NSS database to export the CA certificate:

```
# certutil -d /etc/dirsrv/slapd-instance_name/-L -n "Example CA" -a > /tmp/ds-ca.crt
```

2. Copy the CA certificate to the **/etc/pki/ca-trust/source/anchors/** directory. For example:

```
# cp /tmp/ds-ca.crt /etc/pki/ca-trust/source/anchors/
```

3. Rebuild the CA trust database:

```
# update-ca-trust
```

## 9.5. DISPLAYING THE ENCRYPTION PROTOCOLS ENABLED IN DIRECTORY SERVER

To display the enabled encryption protocols in Directory Server:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security get
...
sslversionmin: TLS1.2
sslversionmax: TLS1.3
```

The **`sslVersionMin`** and **`sslVersionMax`** parameter control which encryption protocol versions Directory Server uses. The default of **`sslVersionMin`** depends on the system-wide crypto policy you use.

## 9.6. SETTING THE MINIMUM TLS ENCRYPTION PROTOCOL VERSION

By default, Directory Server sets **`sslVersionMin`** parameter automatically based on the system-wide crypto policy. The following table provides an overview of the TLS version in **`sslVersionMin`** Directory Server uses based on the system-wide crypto policy profile:

**Table 9.2. Overview of System-wide Crypto Policy Profiles and the Minimum TLS Version They Define**

Profile	Minimum TLS Version
<b>DEFAULT</b>	TLS 1.2
<b>FUTURE</b>	TLS 1.2
<b>FIPS</b>	TLS 1.2
<b>LEGACY</b>	TLS 1.0

For further details about system-wide crypto policy, how to change the profile, and opting-out services of system-wide crypto policies, see the [Using system-wide cryptographic policies](#) section in the *RHEL 8 Security Hardening* guide.

Alternatively, you can manually set **`sslVersionMin`** to higher value than the one defined in the crypto policy profile:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security set --tls-protocol-min="TLS1.3"
```

## 9.7. SETTING THE HIGHEST TLS ENCRYPTION PROTOCOL VERSION

To set the highest TLS protocol version Directory Server supports, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security set --tls-protocol-max="protocol_version"
```

If you set the parameter to a value lower than in **`sslVersionMin`**, then Directory Server sets **`sslVersionMax`** to the same value as **`sslVersionMin`**.



### IMPORTANT

To always use the strongest supported encryption protocol version in the **`sslVersionMax`** parameter, do not set this parameter.

## 9.8. USING HARDWARE SECURITY MODULES

A security module serves as a medium between the Directory Server and the TLS layer. The module stores the keys and certificates used for encryption and decryption. The standard which defines these modules is Public Key Cryptography Standard (PKCS) #11, so these modules are PKCS#11 modules.

By default, Directory Server uses built-in security databases, **key4.db** and **cert9.db**, to store the keys and certificates used by the servers.

It is also possible to use external security devices to store Directory Server certificates and keys. For Directory Server to use an external PKCS#11 module, the module's drivers must be installed in Directory Server.

For more information, consult the documentation for your hardware security module.

## 9.9. USING CERTIFICATE-BASED CLIENT AUTHENTICATION

Directory Server supports certificate-based authentication of LDAP clients and for server-to-server connection, such as replication.

Depending on the configuration, the client can or must authenticate using a certificate, if you enabled certificate-based authentication. After verifying the certificate, the server searches for the user in the directory, based on the attributes in the **subject** field of the certificate. If the search return exactly one user entry, Directory Server uses this user for all further operations. Optionally, you can configure that the certificate used for authentication must match the Distinguished Encoding Rules (DER)-formatted certificate stored in the **userCertificate** attribute of the user.

Benefits of using certificate-based authentication:

- Improved efficiency. When using applications that prompt once for the certificate database password and then use that certificate for all subsequent bind or authentication operations, it is more efficient than continuously providing a bind DN and password.
- Improved security. The use of certificate-based authentication is more secure than non-certificate bind operations because certificate-based authentication uses public-key cryptography. Bind credentials cannot be intercepted across the network. If the certificate or device is lost, it is useless without the PIN, so it is immune from third-party interference like phishing attacks.

### 9.9.1. Setting up Certificate-based Authentication

To enable certificate-based authentication:

1. Enable encrypted connections. For details, see [Section 9.4, "Enabling TLS"](#).
2. Install the CA certificate and set the trust options for client and server connections. See [Section 9.3.2, "Installing a CA Certificate"](#).
3. Optionally, verify that the **CT,,** trust options for client and server are set for the CA certificate:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate get  
"Example-CA"  
Certificate Name: Example-CA  
Subject DN: CN=server.example.com,ST=Queensland,C=AU  
Issuer DN: CN=server.example.com,,ST=Queensland,C=AU  
Expires: 2021-05-09 10:57:54  
Trust Flags: CT,,
```

4. Create the `/etc/dirsrv/slappd-instance_name/certmap.conf` file to map information from the certificate to Directory Server users. For example:

```

certmap default      default
default:DNC parms   dc
default:FilterComps mail,cn
default:VerifyCert   on

certmap example      o=Example Inc.,c=US
example:DNC parms

```

This configures that for authenticating users who use a certificate that has the **o=Example Inc.,c=US** issuer Distinguished Name (DN) set, Directory Server does not generate a base DN from the subject of the certificate, because the **DNC parms** parameter is set empty for this issuer. Additionally, the settings for the **FilterComps** and **VerifyCert** are inherited from the default entry.

Certificates that have a different issuer DN than the specified one will use the settings from the **default** entry and generate the base DN based on the **cn** attributes in the subject of the certificate. This enables Directory Server to start the search under a specific DN, without searching the whole directory.

For all certificates, Directory Server generates the search filter using the **mail** and the **cn** attribute from the certificate's subject. However, if the **mail** does not exist in the subject, Directory Server will automatically use the value of the certificate's **e** attribute in the subject.

For further details and descriptions of the available parameters, see the description of the **certmap.conf** file in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

5. Enable client authentication. For example, to configure that client authentication is optional:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com security set --tls-client-auth="allowed"
```

Alternatively, set the **--tls-client-auth** parameter to **required** to configure that clients must use a certificate to authenticate.

6. If you enabled that the authenticating certificate must match the one stored in the **userCertificate** attribute of the user by setting **alias\_name:VerifyCert on** in the `/etc/dirsrv/slappd-instance_name/certmap.conf` file, add the certificates to the user entries. See [Section 9.9.2, “Adding a Certificate to a User”](#).

### 9.9.2. Adding a Certificate to a User

When you set up certificate-based authentication, you can set that the certificate used to authenticate must match the one stored in the **userCertificate** binary attribute of the user. If you enabled this feature by setting **alias\_name:VerifyCert on** in the `/etc/dirsrv/slappd-instance_name/certmap.conf` file, you must add the certificate of the affected users to their directory entry.



#### IMPORTANT

You must store the certificate in the Distinguished Encoding Rules (DER) format in the **userCertificate** attribute.

To store a certificate in the ***userCertificate*** attribute of a user:

1. If the certificate is not DER-formatted, convert it. For example:

```
# openssl x509 -in /root/certificate.pem -out /root/certificate.der -outform DER
```

2. Add the certificate to the user's ***userCertificate*** attribute. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user_name,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate
userCertificate:< file:///root/example.der
```

For further details about using binary attributes, see [Section 3.8, “Using Binary Attributes”](#).

### 9.9.3. Forcing the EXTERNAL SASL Mechanism for Bind Requests

At the beginning of a TLS session, the client sends its certificate to the server. Then, it sends its bind request. Most clients issue the bind request using the **EXTERNAL** SASL mechanism, which signals Directory Server that it needs to use the identity in the certificate for the bind, instead of the credentials in the bind request.

However, if a client uses simple authentication or anonymous credentials, this information is missing. In this case, the TLS session fails with invalid credentials, even if the certificate and the client identity in the certificate was valid.

To configure that Directory Server forces clients to use the **EXTERNAL** SASL mechanism and to ignore any other bind method in the request:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-force-sasl-external=on
Successfully replaced "nsslapd-force-sasl-external"
```

### 9.9.4. Authenticating Using a Certificate

To use the OpenLDAP client tools, to authenticate to a Directory Server instance that supports authentication using a certificate:

1. Set the following environment variables to the corresponding paths for the CA certificate, the user key, and the user certificate. For example:

```
LDAPTLS_CACERT=/home/user_name/CA.crt
LDAPTLS_KEY=/home/user_name/user.key
LDAPTLS_CERT=/home/user_name/user.crt
```

Alternatively, set the **TLS\_CACERT**, **TLS\_KEY**, and **TLS\_CERT** parameters in the **~/.ldaprc** file. For details, see the **TLS OPTIONS** section in the **ldap.conf(5)** man page.

2. Connect to the server. For example:

```
# ldapwhoami -H ldaps://server.example.com:636
```

If you use a different client, see the client application's documentation for how to connect using certificate-based authentication.

## 9.10. SETTING UP SASL IDENTITY MAPPING

*Simple Authentication and Security Layer* (SASL) is an abstraction layer between protocols like LDAP and authentication methods like GSS-API which allows any protocol which can interact with SASL to utilize any authentication mechanism which can work with SASL. Simply put, SASL is an intermediary that makes authenticating to applications using different mechanisms easier. SASL can also be used to establish an encrypted session between a client and server.

The SASL framework allows different mechanisms to be used to authenticate a user to the server, depending on what mechanism is enabled in both client and server applications. SASL also creates a layer for encrypted (secure) sessions. Using GSS-API, Directory Server utilizes Kerberos tickets to authenticate sessions and encrypt data.

### 9.10.1. About SASL Identity Mapping

When processing a SASL bind request, the server matches, or maps, the SASL authentication ID used to authenticate to the Directory Server with an LDAP entry stored within the server. When using Kerberos, the SASL user ID usually has the format *userid@REALM*, such as **scarter@EXAMPLE.COM**. This ID must be converted into the DN of the user's Directory Server entry, such as  
**uid=scarter,ou=people,dc=example,dc=com**.

If the authentication ID clearly corresponds to the LDAP entry for a person, it is possible to configure the Directory Server to map the authentication ID automatically to the entry DN. Directory Server has some pre-configured default mappings which handle most common configurations, and customized maps can be created. By default, during a bind attempt, only the first matching mapping rule is applied if SASL mapping fallback is not enabled. For further details about SASL mapping fallback, see [Section 9.10.4, "Enabling SASL Mapping Fallback"](#).

Be sure to configure SASL maps so that only one mapping rule matches the authentication string.

SASL mappings are configured by entries under a container entry:

```
dn: cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: sasl
```

SASL identity mapping entries are children of this entry:

```
dn: cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: mapping
```

Mapping entries are defined by the following attributes:

- **nsSaslMapRegexString**: The regular expression which is used to map the elements of the supplied **authid**.
- **nsSaslMapFilterTemplate**: A template which applies the elements of the **nsSaslMapRegexString** to create the DN.

- ***nsSaslMapBaseDNTemplate***: Provides the search base or a specific entry DN to match against the constructed DN.
- Optional: ***nsSaslMapPriority***: Sets the priority of this SASL mapping. The priority value is used, if ***nsslapd-sasl-mapping-fallback*** is enabled in ***cn=config***. For details, see [Section 9.10.4.1, "Setting SASL Mapping Priorities"](#).

For further details, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

For example:

```
dn: cn=mymap,cn=mapping,cn=sasl,cn=config
objectclass:top
objectclass:nsSaslMapping
cn: mymap
nsSaslMapRegexString: \(.*\)@(.*)\.(.*)
nsSaslMapFilterTemplate: (objectclass=inetOrgPerson)
nsSaslMapBaseDNTemplate: uid=\1,ou=people,dc=\2,dc=\3
```

The ***nsSaslMapRegexString*** attribute sets variables of the form **\1**, **\2**, **\3** for bind IDs which are filled into the template attributes during a search. This example sets up a SASL identity mapping for any user in the **ou=People,dc=example,dc=com** subtree who belongs to the **inetOrgPerson** object class.

When a Directory Server receives a SASL bind request with **mconnors@EXAMPLE.COM** as the user ID (**authid**), the regular expression fills in the base DN template with **uid=mconnors,ou=people,dc=EXAMPLE,dc=COM** as the user ID, and authentication proceeds from there.



#### NOTE

The **dc** values are not case sensitive, so **dc=EXAMPLE** and **dc=example** are equivalent.

The Directory Server can also use a more inclusive mapping scheme, such as the following:

```
dn: cn=example map,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: example map
nsSaslMapRegexString: \(.*\)
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This matches any user ID and map it an entry under the **ou=People,dc=example,dc=com** subtree which meets the filter **cn=userId**.

Mappings can be confined to a single realm by specifying the realm in the ***nsSaslMapRegexString*** attribute. For example:

```
dn: cn=example map,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: example map
```

```
nsSaslMapRegexString: \(.*)@US.EXAMPLE.COM
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This mapping is identical to the previous mapping, except that it only applies to users authenticating from the **US.EXAMPLE.COM** realm. (Realms are described in [Section 9.11.2.1, "About Principals and Realms"](#).)

When a server connects to another server, such as during replication or with chaining, the default mappings for the will not properly map the identities. This is because the principal (SASL identity) for one server does not match the principal on the server where authentication is taking place, so it does not match the mapping entries.

To allow server to server authentication using SASL, create a mapping for the specific server principal to a specific user entry. For example, this mapping matches the **ldap1.example.com** server to the **cn=replication manager,cn=config** entry. The mapping entry itself is created on the second server, such as **ldap2.example.com**.

```
dn: cn=z,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: z
nsSaslMapRegexString: ldap/ldap1.example.com@EXAMPLE.COM
nsSaslMapBaseDNTemplate: cn=replication manager,cn=config
nsSaslMapFilterTemplate: (objectclass=*)
```

Sometimes, the realm name is not included in the principal name in SASL GSS-API configuration. A second mapping can be created which is identical to the first, only without specifying the realm in the principal name. For example:

```
dn: cn=y,cn=mapping,cn=sasl,cn=config
objectclass: top
objectclass: nsSaslMapping
cn: y
nsSaslMapRegexString: ldap/ldap1.example.com
nsSaslMapBaseDNTemplate: cn=replication manager,cn=config
nsSaslMapFilterTemplate: (objectclass=*)
```

Because the realm is not specified, the second mapping is more general (meaning, it has the potential to match more entries than the first). The best practice is to have more specific mappings processed first and gradually progress through more general mappings.

If a priority is not set for a SASL mapping using the **nsSaslMapPriority** parameter, there is no way to specify the order that mappings are processed. However, there is a way to control how SASL mappings are processed: the name. The Directory Server processes SASL mappings in reverse ASCII order. In the past two example, then the **cn=z** mapping (the first example) is processed first. If there is no match, the server processes the **cn=y** mapping (the second example).



## NOTE

SASL mappings can be added when an instance is created during a silent installation by specifying the mappings in an LDIF file and adding the LDIF file with the **ConfigFile** directive. Using silent installation is described in the *Installation Guide*.

## 9.10.2. Default SASL Mappings for Directory Server

The Directory Server has pre-defined SASL mapping rules to handle some of the most common usage.

### Kerberos UID Mapping

This matches a Kerberos principal using a two part realm, such as `user@example.com`. The realm is then used to define the search base, and the user ID (***authid***) defines the filter. The search base is `dc=example,dc=com` and the filter of (***uid=user***).

```
dn: cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: Kerberos uid mapping
nsSaslMapRegexString: \(.*\)@(.*)\.(.*)
nsSaslMapBaseDNTemplate: dc=\2,dc=\3
nsSaslMapFilterTemplate: (uid=\1)
```

### RFC 2829 DN Syntax

This mapping matches an ***authid*** that is a valid DN (defined in RFC 2829) prefixed by `dn:`. The ***authid*** maps directly to the specified DN.

```
dn: cn=rfc 2829 dn syntax,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: rfc 2829 dn syntax
nsSaslMapRegexString: ^dn:(.*)
nsSaslMapBaseDNTemplate: \1
nsSaslMapFilterTemplate: (objectclass=*)
```

### RFC 2829 U Syntax

This mapping matches an ***authid*** that is a UID prefixed by `u:`. The value specified after the prefix defines a filter of (***uid=value***). The search base is hard-coded to be the suffix of the default ***userRoot*** database.

```
dn: cn=rfc 2829 u syntax,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: rfc 2829 u syntax
nsSaslMapRegexString: ^u:(.*)
nsSaslMapBaseDNTemplate: dc=example,dc=com
nsSaslMapFilterTemplate: (uid=\1)
```

### UID Mapping

This mapping matches an ***authid*** that is any plain string that does not match the other default mapping rules. It uses this value to define a filter of (***uid=value***). The search base is hard-coded to be the suffix of the default ***userRoot*** database.

```
dn: cn=uid mapping,cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsSaslMapping
cn: uid mapping
```

```
nsSaslMapRegexString: ^[^:@]+$  
nsSaslMapBaseDNTemplate: dc=example,dc=com  
nsSaslMapFilterTemplate: (uid=&)
```

### 9.10.3. Configuring SASL Identity Mapping

(Simple Authentication and Security Layer) SASL identity mapping can be configured from either the Directory Server or the command line. For SASL identity mapping to work for SASL authentication, the mapping must return one, and only one, entry that matches and Kerberos must be configured on the host machine.

#### 9.10.3.1. Configuring SASL Identity Mapping Using the Command Line

To configure SASL identity mapping from the command line, use the **dsconf** utility to add the identity mapping scheme.

1. Add the identity mapping scheme. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com sasl create --cn  
"example_map" --nsSaslMapRegexString "|(.*)|" --nsSaslMapBaseDNTemplate  
"ou=People,dc=example,dc=com" --nsSaslMapFilterTemplate "(cn=|1)" --nsSaslMapPriority  
50  
Successfully created example_map
```

This matches any user's common name and maps it to the result of the subtree search with base **ou=People,dc=example,dc=com**, based on the filter **cn=userId**.

2. Restart the instance:

```
# dsctl instance_name restart
```



#### NOTE

Adding the SASL map with **dsconf** adds the mapping to the end of the list, regardless of its ASCII order.

#### 9.10.3.2. Configuring SASL Identity Mapping Using the Web Console

To add a SASL identity mapping scheme:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select **SASL Settings & Mappings**.
4. Click **Create New Mapping**.
5. Fill the form. For example:

**Create SASL Mapping**

SASL Mapping Name	Example Mapping	
SASL Mapping Regex	\(.*\)	
* Test Regex	Enter text to test regex	Test It
SASL Mapping Base	ou=Users,dc=example,dc=com	
SASL Mapping Filter	(objectClass=person)	
SASL Mapping Priority	100	
<input type="button" value="Cancel"/> <input type="button" value="Create Mapping"/>		

6. Click **Save**.

#### 9.10.4. Enabling SASL Mapping Fallback

Using the default settings, Directory Server verifies only the first matching SASL mapping. If this first matching mapping fails, the bind operation fails and no further matching mappings are verified.

However, you can configure Directory Server to verify all matching mappings by enabling the ***nsslapd-sasl-mapping-fallback*** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-sasl-mapping-fallback=on
Successfully replaced "nsslapd-sasl-mapping-fallback"
```

If fallback is enabled and only one user identity is returned, the bind succeeds. If no user, or more than one user is returned, the bind fails.

##### 9.10.4.1. Setting SASL Mapping Priorities

If you enabled SASL mapping fallback using the ***nsslapd-sasl-mapping-fallback*** attribute, you can optionally set the ***nsSaslMapPriority*** attribute in mapping configurations to prioritize them. The ***nsSaslMapPriority*** attribute supports values from **1** (highest priority) to **100** (lowest priority). The default is **100**.

For example, to set the highest priority for the **cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config** mapping:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config
changetype: modify
replace: nsSaslMapPriority
nsSaslMapPriority: 1
```

## 9.11. USING KERBEROS GSS-API WITH SASL

Kerberos v5 must be deployed on the host for Directory Server to utilize the GSS-API mechanism for SASL authentication. GSS-API and Kerberos client libraries must be installed on the Directory Server host to take advantage of Kerberos services.

### 9.11.1. Authentication Mechanisms for SASL in Directory Server

Directory Server support the following SASL encryption mechanisms:

- *PLAIN*. PLAIN sends cleartext passwords for simple password-based authentication.
- *EXTERNAL*. EXTERNAL, as with TLS, performs certificate-based authentication. This method uses public keys for strong authentication.
- *CRAM-MD5*. **CRAM-MD5** is a weak, simple challenge-response authentication method. It does not establish any security layer.



#### WARNING

Red Hat recommends not using the insecure **CRAM-MD5** mechanism.

- *DIGEST-MD5*. **DIGEST-MD5** is a weak authentication method for LDAPv3 servers.



#### WARNING

Red Hat recommends not using the insecure **DIGEST-MD5** mechanism.

- *Generic Security Services (GSS-API)*. Generic Security Services (GSS) is a security API that is the native way for UNIX-based operating systems to access and authenticate Kerberos services. GSS-API also supports session encryption, similar to TLS. This allows LDAP clients to authenticate with the server using Kerberos version 5 credentials (tickets) and to use network session encryption.

For Directory Server to use GSS-API, Kerberos must be configured on the host machine. See [Section 9.11, “Using Kerberos GSS-API with SASL”](#).



#### NOTE

GSS-API and, thus, Kerberos are only supported on platforms that have GSS-API support. To use GSS-API, it may be necessary to install the Kerberos client libraries; any required Kerberos libraries will be available through the operating system vendor.

## 9.11.2. About Kerberos in Directory Server

On Red Hat Enterprise Linux, the supported Kerberos libraries are MIT Kerberos version 5.

The concepts of Kerberos, as well as using and configuring Kerberos, are covered at the MIT Kerberos website, <http://web.mit.edu/Kerberos/>.

### 9.11.2.1. About Principals and Realms

A *principal* is a user or service in the Kerberos environment. A *realm* defines what Kerberos manages in terms of who can access what. The client, the KDC, and the host or service you want to access must use the same realm.



#### NOTE

Kerberos realms are only supported for GSS-API authentication and encryption, not for DIGEST-MD5.

Realms are used by the server to associate the DN of the client in the following form, which looks like an LDAP DN:

```
uid=user_name/[server_instance],cn=realm,cn=mechanism,cn=auth
```

For example, Mike Connors in the **engineering** realm of the European division of **example.com** uses the following association to access a server in the US realm:

```
uid=mconnors/cn=Europe.example.com,cn=engineering,cn=gssapi,cn=auth
```

Babara Jensen, from the **accounting** realm of **US.example.com**, does not have to specify a realm when to access a local server:

```
uid=bjensen,cn=accounting,cn=gssapi,cn=auth
```

If realms are supported by the mechanism and the default realm is not used to authenticate to the server, then the *realm* must be specified in the Kerberos principal. Otherwise, the realm can be omitted.



#### NOTE

Kerberos systems treat the Kerberos realm as the default realm; other systems default to the server.

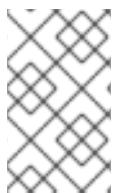
### 9.11.2.2. About the KDC Server and Keytabs

The Key Distribution Center (KDC) authenticates users and issues Ticket Granting Tickets (TGT) for them. This enables users to authenticate to Directory Server using GSS-API. To respond to Kerberos operations, Directory Server requires access to its keytab file. The keytab contains the cryptographic key that Directory Server uses to authenticate to other servers.

Directory Server uses the **ldap** service name in a Kerberos principal. For example:

```
ldap/server.example.com/EXAMPLE.COM
```

For details about creating the keytab, see your Kerberos documentation.



#### NOTE

You must create a Simple Authentication and Security Layer (SASL) mapping for the Directory Server Kerberos principal that maps to an existing entry Distinguished Name (DN).

### 9.11.3. Configuring SASL Authentication at Directory Server Startup

SASL GSS-API authentication has to be activated in Directory Server so that Kerberos tickets can be used for authentication. This is done by supplying a system configuration file for the init scripts to use which identifies the variable to set the keytab file location. When the init script runs at Directory Server startup, SASL authentication is then immediately active.

The default SASL configuration is stored in the `/etc/sysconfig/dirsrv` file.

If there are multiple Directory Server instances and not all of them will use SASL authentication, then there can be instance-specific configuration files created in the `/etc/sysconfig/` directory named `dirsrv-instance`. For example, `dirsrv-example`. The default `dirsrv` file can be used if there is a single instance on a host.

To enable SASL authentication, uncomment the `KRB5_KTNAME` line in the `/etc/sysconfig/dirsrv` (or instance-specific) file, and set the keytab location for the `KRB5_KTNAME` variable. For example:

```
# In order to use SASL/GSSAPI the directory
# server needs to know where to find its keytab
# file - uncomment the following line and set
# the path and filename appropriately
KRB5_KTNAME=/etc/dirsrv/krb5.keytab
```

## 9.12. SETTING SASL MECHANISMS

Per default, Directory Server enables all mechanisms the simple authentication and security layer (SASL) library supports. These are listed in the root `dse` `supportedSASLMechanisms` parameter. To enable specific SASL mechanisms, set the `nsslapd-allowed-sasl-mechanisms` attribute in the `cn=config` entry. For example, to enable only the `GSSAPI` and `DIGEST-MD5` mechanism, run:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-allowed-sasl-
mechanisms="GSSAPI, DIGEST-MD5"
Successfully replaced "nsslapd-allowed-sasl-mechanisms"
```



#### NOTE

Even if `EXTERNAL` is not listed in the `nsslapd-allowed-sasl-mechanisms` parameter, this mechanism is always enabled.

For further details, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

## 9.13. USING SASL WITH LDAP CLIENTS

To use SASL with the LDAP clients, such as **ldapsearch**, pass the **-Y SASL\_mechanism** to the command. For example:

- To use the **GSSAPI** SASL mechanism over the LDAP protocol:

```
# ldapsearch -Y GSSAPI -U "dn:uid=user_name,ou=people,dc=example,dc=com" -R EXAMPLE.COM -H ldap://server.example.com -b "dc=example,dc=com"
```

- To use the **PLAIN** SASL mechanism over the LDAPS protocol:

```
# ldapsearch -Y PLAIN -D "uid=user_name,ou=people,dc=example,dc=com" -W -H ldaps://server.example.com -b "dc=example,dc=com"
```



#### NOTE

SASL proxy authorization is not supported in Directory Server. Therefore, Directory Server ignores any SASL **authzid** value supplied by the client.

# CHAPTER 10. CONFIGURING ATTRIBUTE ENCRYPTION

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

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

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

An additional benefit of attribute encryption is, that encrypted values can only be sent to a clients with a Security Strength Factor (SSF) greater than 1.



## NOTE

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

```
dn: uid=jsmith1234,ou=People,dc=example,dc=com
...
uid:: Sf04P9nJWGU1qiW9JJCGRg==
```

That would allow someone to discover the encrypted value.

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

Indexed attributes may be encrypted, and attribute encryption is fully compatible with **eq** and **pres** 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.

## 10.1. ENCRYPTION KEYS

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

Randomly generated symmetric cipher keys are used to encrypt and decrypt attribute data. A separate

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



### WARNING

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



### WARNING

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

## 10.2. ENCRYPTION CIPHERS

The encryption cipher is configurable on a per-attribute basis and must be selected by the administrator at the time encryption is enabled for an attribute.

The following ciphers are supported:

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



### NOTE

For strong encryption, Red Hat recommends using only AES ciphers.

All ciphers are used in Cipher Block Chaining mode.

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

## 10.3. CONFIGURING ATTRIBUTE ENCRYPTION

Use the command line or the web console to enable and disable attribute encryption for certain attributes.

### 10.3.1. Enabling Encryption of an Attribute Using the Command Line

To configure that Directory Server stores, for example, **telephoneNumber** attributes in the **userRoot** database AES-encrypted:

1. Optionally, to encrypt existing **telephoneNumber** attributes, export the database. See [Section 10.4.1, “Exporting an Encrypted Database”](#).
2. Enable AES encryption for the **telephoneNumber** attribute in the **userRoot** database:

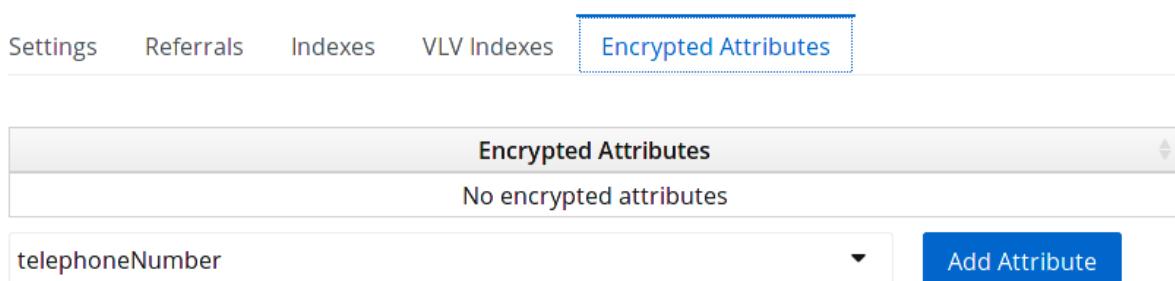
```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend attr-encrypt --add-attr telephoneNumber userRoot
```

3. If you exported the database to encrypt also existing attributes, reimport the database. See [Section 10.4.2, “Importing an LDIF File into an Encrypted Database”](#).

### 10.3.2. Enabling Encryption of an Attribute Using the Web Console

To configure that Directory Server stores, for example, **telephoneNumber** attributes in the database AES-encrypted:

1. Optionally, to encrypt existing **telephoneNumber** attributes, export the database. See [Section 10.4.1, “Exporting an Encrypted Database”](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. Open the **Database** menu.
5. Select the suffix entry.
6. Open the **Encrypted Attributes** tab.
7. Enter the name of the attribute to be encrypted.



8. Click **Add Attribute**.
9. If you exported the database to encrypt also existing attributes, reimport the database. See [Section 10.4.2, “Importing an LDIF File into an Encrypted Database”](#).

### 10.3.3. Disabling Encryption for an Attribute Using the Command Line

To configure that Directory Server no longer stores, for example, **telephoneNumber** attributes encrypted in the **userRoot** database:

1. Optionally, to decrypt existing **telephoneNumber** attributes, export the database. See [Section 10.4.1, "Exporting an Encrypted Database"](#).
2. Disable encryption for the **telephoneNumber** attribute in the **userRoot** database:

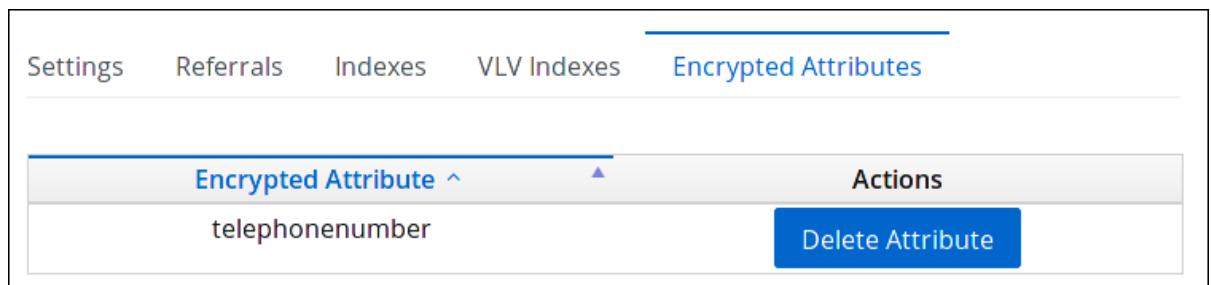
```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend attr-encrypt --del-attr telephoneNumber userRoot
```

3. If you exported the database to decrypt existing attributes, reimport the database. See [Section 10.4.2, "Importing an LDIF File into an Encrypted Database"](#).

#### 10.3.4. Disabling Encryption of an Attribute Using the Web Console

To configure that Directory Server stores, for example, **telephoneNumber** attributes in the database AES-encrypted:

1. Optionally, to encrypt existing **telephoneNumber** attributes, export the database. See [Section 10.4.1, "Exporting an Encrypted Database"](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
3. Select the instance.
4. Open the **Database** menu.
5. Select the suffix entry.
6. Open the **Encrypted Attributes** tab.
7. Click the **Delete Attribute** button to the right of the **telephoneNumber** attribute.



8. Click **Yes** to confirm.
9. If you exported the database to decrypt existing attributes, reimport the database. See [Section 10.4.2, "Importing an LDIF File into an Encrypted Database"](#).

#### 10.3.5. General Considerations after Enabling Attribute Encryption

When you enabled encryption for data that is already in the database:

- Unencrypted data can persist in the server's database page pool backing file. To remove this data:
  1. Stop the instance:

```
# dsctl instance_name stop
```

2. Delete the `/var/lib/dirsrv/slapd-instance_name/db/guardian` file:

```
# rm /var/lib/dirsrv/slapd-instance_name/db/guardian
```

3. Start the instance:

```
# dsctl instance_name start
```

- After you enabled encryption and successfully imported the data, delete the LDIF file with the unencrypted data.
- After enabling encryption, Directory Server deletes and creates a new database when reimporting the data.
- The replication log file is not encrypted. To protect this data, store it on an encrypted disk.
- Data in the server's memory (RAM) is unencrypted and can be temporarily stored in swap partitions. To protect this data, set up encrypted swap space.



### IMPORTANT

Even if you delete files that contain unencrypted data, this data can be restored under certain circumstances.

## 10.4. EXPORTING AND IMPORTING AN ENCRYPTED DATABASE

Exporting and importing encrypted databases is similar to exporting and importing regular databases. However, the encrypted information must be decrypted when you export the data and re-encrypted when you reimport it to the database.

### 10.4.1. Exporting an Encrypted Database

To export data from an encrypted database, pass the `-E` parameter to the `dsconf` command.

For example, to export the complete `userRoot` database with decrypted attributes:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export -E userRoot
```

Alternatively, you can export only a specific subtree. For example, to export all data from the `ou=People,dc=example,dc=com` entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export -E -s
"ou=People,dc=example,dc=com" userRoot
```

For further details about using `dsconf` to export data, see [Section 6.2.1.1.1, “Exporting a Databases Using the `dsconf backend export` Command”](#).

### 10.4.2. Importing an LDIF File into an Encrypted Database

To import data to a database when attribute encryption is enabled:

- Stop the Directory Server instance:

```
# dsctl instance_name stop
```

2. If you replaced the certificate database between the last export and this import, edit the **/etc/dirsrv/slappd-*instance\_name*/dse.ldif** file, and remove the following entries including their attributes:

- **cn=AES,cn=encrypted attribute keys,cn=database\_name,cn=ldbm database,cn=plugins,cn=config**
- **cn=3DES,cn=encrypted attribute keys,cn=database\_name,cn=ldbm database,cn=plugins,cn=config**



### IMPORTANT

Remove the entries for all databases. If any entry that contains the **nsSymmetricKey** attribute is left in the **/etc/dirsrv/slappd-*instance\_name*/dse.ldif** file, Directory Server will fail to start.

3. Import the LDIF file. For example, to import the **/tmp/example.ldif** into the **userRoot** database:

```
# dsctl instance_name ldif2db --encrypted userRoot /tmp/example.ldif
```

The **--encrypted** parameter enables the script to encrypt attributes configure for encryption during the import.

4. Start the instance:

```
# dsctl instance_name start
```

## 10.5. UPDATING THE TLS CERTIFICATES USED FOR ATTRIBUTE ENCRYPTION

Attribute encryption is based on the TLS certificate. To prevent that attribute encryption fails after renewing or replacing the TLS certificate:

1. Export the database with decrypted attributes. See [Section 10.4.1, “Exporting an Encrypted Database”](#).
2. Delete the existing private key and certificate from the Network Security Services (NSS) database. See [Section 9.3.8, “Removing a Private Key”](#)
3. Create a new Certificate Signing Request (CSR). See [Section 9.3.1, “Creating a Certificate Signing Request”](#).
4. Install the new certificate. See [Section 9.3.4, “Installing a Server Certificate”](#).
5. Stop the Directory Server instance:

```
# dsctl instance_name stop
```

6. Edit the **/etc/dirsrv/slappd-*instance\_name*/dse.ldif** file and remove the following entries including their attributes:

- **cn=AES,cn=encrypted attribute keys,cn=database\_name,cn=ldbm database,cn=plugins,cn=config**
- **cn=3DES,cn=encrypted attribute keys,cn=database\_name,cn=ldbm database,cn=plugins,cn=config**



### IMPORTANT

Remove the entries for all databases. If any entry that contains the **nsSymmetricKey** attribute is left in the **/etc/dirsrv/slappd-*instance\_name*/dse.ldif** file, Directory Server will fail to start.

7. Import the database. See [Section 10.4.2, “Importing an LDIF File into an Encrypted Database”](#).

8. Start the instance:

```
# dsctl instance_name start
```

# CHAPTER 11. MANAGING FIPS MODE SUPPORT

Red Hat Directory Server fully supports the Federal Information Processing Standard (FIPS) 140-2. When Directory Server runs in FIPS mode, security-related settings change. For example, SSL is automatically disabled and only TLS 1.1 and 1.2 encryption is used.

For general details about FIPS, see [Federal Information Processing Standard \(FIPS\)](#) in the *Red Hat Enterprise Linux Security Guide*.

## Enabling FIPS Mode Support

To enable FIPS mode support for Directory Server:

1. Optionally, enable FIPS mode in Red Hat Enterprise Linux. For details, see the corresponding section in the [Red Hat Enterprise Linux Security Guide](#).
2. Enable FIPS mode for the network security services (NSS) database:

```
# modutil -dbdir /etc/dirsrv/slapd-instance_name -fips true
```

3. Restart the Directory Server instance:

```
# dsctl instance_name restart
```

## Disabling FIPS Mode Support

To disable FIPS mode support for Directory Server:

1. Disable FIPS mode for the network security services (NSS) database:
2. Restart the Directory Server instance:
3. Optionally, disable FIPS mode in Red Hat Enterprise Linux. For details, see the corresponding section in the [Red Hat Enterprise Linux Security Guide](#).

# CHAPTER 12. MANAGING THE DIRECTORY SCHEMA

Red Hat Directory Server comes with a standard schema that includes hundreds of object classes and attributes. While the standard object classes and attributes should meet most deployments' requirements, it can be necessary to extend the schema for specific directory data. Extending the schema is done by creating new object classes and attributes.

The [Red Hat Directory Server 11 Configuration, Command, and File Reference](#) is a reference for most the standard Directory Server attributes and object classes, with information on allowed and required attributes, which object classes take which attribute, and OID and value information. This is a good resource for identifying useful schema elements for a directory and determining what custom schema needs to be created.

## 12.1. OVERVIEW OF SCHEMA

The directory schema is a set of rules that defines how data can be stored in the directory. Directory information is stored discrete entries, and each entry is comprised of a set of attributes and their values. The kind of identity being described in the entry is defined in the entry's object classes. An object class specifies the kind of object the entry describes through the defined set of attributes for the object class.

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, including people, groups, locations, organizations and divisions, and equipment. The identity is described in a directory entries with attributes and their values, pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. Other aspects of the Directory Server configuration, including matching rules and LDAP controls, are also defined in the schema. All of these together are *schema elements*.

Every schema element is identified by a unique, dot-separated number. This is called the *object identifier* or *OID*.

### 12.1.1. Default Schema Files

The schema for Directory Server is defined in several different schema files (LDIF files which define schema elements). The Directory Server schema files are located in the `/usr/share/dirsrv/schema/` directory. The files in this directory are used as templates for new Directory Server instances. Adding a new schema into this directory will make it available to any new instances.

The attributes used by the Directory Server to perform operations and manage entries is described with other configuration settings in the [Red Hat Directory Server 11 Configuration, Command, and File Reference](#).

### 12.1.2. Object Classes

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, such as people (**person** and **inetOrgPerson**), groups (**groupOfNames**), locations (**locality**), organizations and divisions (**organization** and **organizationalUnit**), and equipment (**device**).

In a schema file, an object class is identified by the **objectclasses** line, then followed by its OID, name, a description, its direct superior object class (an object class which is required to be used in conjunction with the object class and which shares its attributes with this object class), and the list of required (**MUST**) and allowed (**MAY**) attributes.

This is shown in [Example 12.1, “person Object Class Schema Entry”](#).

### Example 12.1. person Object Class Schema Entry

```
objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard LDAP objectclass' SUP top MUST ( sn $ cn ) MAY ( description $ seeAlso $ telephoneNumber $ userPassword ) X-ORIGIN 'RFC 4519' )
```

Every object class defines a number of required attributes (**MUST** keyword in the schema) and of allowed attributes (**MAY** keyword in the schema). Required attributes must be present in entries using the specified object class, while allowed attributes are permissible and available for the entry to use, but are not required for the entry to be valid.

As in [Example 12.1, “person Object Class Schema Entry”](#), the **person** object class requires the ***cn***, ***sn***, and ***objectClass*** attributes and allows the ***description***, ***seeAlso***, ***telephoneNumber***, and ***userPassword*** attributes.

An object class can inherit attributes from another class, in addition to its own required and allowed attributes. The second object class is the *superior* or *parent* object class of the first.

For example, a user's entry has to have the **inetOrgPerson** object class. In that case, the entry must also include the superior object class for **inetOrgPerson**, **organizationalPerson**, and the superior object class for **organizationalPerson**, which is **person**:

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

An object class definition is an ***objectclasses*** attribute for the ***cn=schema*** entry. The ***objectclasses*** attribute has the format:

```
objectclasses: ( definition )
```

The object class definition contains several components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The superior, or parent, object class for this object class, in the form **SUP** *object\_class*; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry
- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs (\$)
- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs (\$)

Customer object class definitions are stored in the `/etc/dirsrv/slappd-instance_name/schema/99user.ldif` when using the command line or the web console to modify `cn=schema` entries.

### 12.1.3. Attributes

Directory entries are composed of attributes and their values. These pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. For instance, the `cn` attribute is used to store a person's full name, such as `cn: John Smith`.

Additional attributes can supply additional information about John Smith:

```
givenname: John
surname: Smith
mail: jsmith@example.com
```

In a schema file, an attribute is described by:

- OID
- name
- syntax matching rule (optional)
- substring matching rules (optional)
- ordering rule (optional)
- description (optional)
- syntax
- single-valued or multi-valued attribute
- details about where the attribute is defined

This is shown in [Example 12.2, “uid Attribute Schema Entry”](#).

#### Example 12.2. uid Attribute Schema Entry

```
( 0.9.2342.19200300.100.1.1 NAME ( 'uid' 'userid' ) EQUALITY caseIgnoreMatch SUBSTR
caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'RFC 4519' )
```

#### 12.1.3.1. Directory Server Attribute Syntaxes

The attribute's syntax defines the format of the values which the attribute allows; as with other schema elements, the syntax is defined for an attribute using the syntax's OID in the schema file entry.

The Directory Server uses the attribute's syntax to perform sorting and pattern matching on entries.

For more information about LDAP attribute syntaxes, see [RFC 4517](#).

Supported LDAP attribute syntaxes are covered in section *Directory Server Attribute Syntaxes* of the [Red Hat Directory Server 10 Configuration, Command, and File Reference](#).

### 12.1.4. Extending the Schema

New, custom attributes and object classes can be added to a Directory Server instance to extend the schema, and there are several ways to add schema elements. Using LDAP tools adds schema elements to the default custom schema file for an instance, **99user.ldif**. It is also possible to create a new, separate schema file and include it with the default schema files.

Adding new schema elements requires three things:

1. Planning and defining OIDs for the new schema. Schema elements are recognized by the server by their OID, so it is important for the OIDs to be unique and organized. Directory Server itself does not manage OIDs, but there are some best practices described in [Section 12.2, “Managing Object Identifiers”](#).
2. Create the new attributes. Attribute definitions require a name, a syntax (the allowed format of the values), an OID, and a description of whether the attribute can only be used once per entry or multiple times.
3. Create an object class to contain the new attributes. An object class lists the required attributes for that entry type and the allowed (permissible) attributes. Because the default schema should never be altered, if any new attributes are created, then they should be added to a custom object class.

The schema elements should be planned in advance; do not use multiple attributes for the same information. Whenever possible, use the standard Directory Server schema. Directory Server has hundreds of attributes and dozens of object classes defined in the default schema files. The [Red Hat Directory Server 11 Configuration, Command, and File Reference](#) lists and describes the standard attributes and object classes; all of the schema can be viewed in the schema files in **/usr/share/dirsrv/schema/**. Become familiar with the available schema; then plan what information attributes are missing and how best to fill those gaps with custom attributes. Planning the schema is covered in the *Deployment Guide*.



#### WARNING

The default object classes and attributes in Directory Server are based on LDAP and X.500 standards and RFCs. Using standard schema makes the Directory Server more easily integrated with other applications and servers and allows interoperability with LDAP clients, legacy Directory Server instances, and future release. It is inadvisable for you to edit the standard attributes or change the object classes.

Keep the following rules in mind when customizing the Directory Server schema:

- Keep the schema as simple as possible.
- Reuse existing schema elements whenever possible.
- Minimize the number of mandatory attributes defined for each object class.
- Do not define more than one object class or attribute for the same purpose.

- Do not modify any existing definitions of attributes or object classes.



### NOTE

Never delete or replace the standard schema. Doing so can lead to compatibility problems with other directories or other LDAP client applications.

The schema is loaded into the Directory Server instance when the instance is started; any new schema files are not loaded until the Directory Server is restarted or unless a reload task is initiated. The default custom schema file for an instance, `99user.ldif`, is loaded as the last schema file. If it contains definitions already present in standard schema files, the custom definition will override the standard ones.

#### 12.1.5. Schema Replication

When the directory schema is updated in the `cn=schema` sub-tree, Directory Server stores the changes in the local `/etc/dirsrv/slappd-instance_name/schema/99user.ldif` file, including a change state number (CSN). The updated schema is not automatically replicated to other replicas. The schema replication starts when directory content is updated in the replicated tree. For example, if you update a user or group entry after modifying the schema, the supplier compares the CSN stored in the `nsSchemaCSN` attribute with the one on the consumer. If the remote CSN is lower than the one on the supplier, the schema is replicated to the consumer. For a successful replication, all object classes and attribute types on the supplier must be a superset of the consumer's definition.

##### Example 12.3. Schema subsets and supersets

- On **server1**, the `demo` object class allows the `a1`, `a2`, and `a3` attributes.
- On **server2**, the `demo` object class allows the `a1` and `a3` attributes.

In [Example 12.3, “Schema subsets and supersets”](#), the schema definition of the `demo` object class on **server1** is a superset of the object class on **server2**. During the validation phase, when the schema is being replicated or accepted, Directory Server retrieves the superset definitions. For example, if a consumer detects that an object class in the local schema allows less attributes than the object class in the supplier schema, the local schema is updated.

If the schema definitions are successfully replicated, the `nsSchemaCSN` attributes are identical on both servers and no longer compared at the beginning of a replication session.

In the following scenarios, the schema is not replicated:

- The schema on one host is a subset of the schema of another host.

For example, in [Example 12.3, “Schema subsets and supersets”](#), the schema definition of the `demo` object class on **server2** is a subset of the object class on **server1**. Subsets can also occur for attributes (a single-value attribute is a subset of a multi-value attribute) and attribute syntaxes (`IA5` is a subset of `Octet_string`).

- When definitions in supplier schema and consumer schema need to be merged.

Directory Server does not support merging schemas. For example, if an object class on one server allows the `a1`, `a2`, and `a3` attributes and `a1`, `a3`, and `a4` on the other, the schemas are not subsets and cannot be merged.

- Schema files other than `/etc/dirsrv/slappd-instance_name/schema/99user.ldif` are used.

Directory Server enables you to add additional schema files in the `/etc/dirsrv/slappd-instance_name/schema` directory. However, only the CSN in the `99user.ldif` file is updated. For this reasons, other schema file are only used locally and are not automatically transferred to replication partners. Copy the updated schema file manually to the consumers and reload the schema. For details, see [Section 12.10, “Dynamically Reloading Schema”](#).

To avoid duplicate schema definitions and to enable automatic replication, store all custom schema in the `/etc/dirsrv/slappd-instance_name/schema/99user.ldif` file. For further information about creating custom schema files, see [Section 12.9, “Creating Custom Schema Files”](#).

## 12.2. MANAGING OBJECT IDENTIFIERS

Each LDAP object class or attribute must be assigned a unique name and *object identifier* (OID). An OID is a dot-separated number which identifies the schema element to the server. OIDs can be hierarchical, with a base OID that can be expanded to accommodate different branches. For example, the base OID could be **1**, and there can be a branch for attributes at **1.1** and for object classes at **1.2**.



### NOTE

It is not required to have a numeric OID for creating custom schema, but Red Hat strongly recommends it for better forward compatibility and performance.

OIDs are assigned to an organization through the Internet Assigned Numbers Authority (IANA), and Directory Server does not provide a mechanism to obtain OIDs. To get information about obtaining OIDs, visit the IANA website at <http://www.iana.org/cgi-bin/enterprise.pl>.

After obtaining a base OID from IANA, plan how the OIDs are going to be assigned to custom schema elements. Define a branch for both attributes and object classes; there can also be branches for matching rules and LDAP controls.

Once the OID branches are defined, create an OID registry to track OID assignments. An OID registry is a list that gives the OIDs and descriptions of the OIDs used in the directory schema. This ensures that no OID is ever used for more than one purpose. Publish the OID registry with the custom schema.

## 12.3. CREATING AN OBJECT CLASS

This section describes how to create a object class using the command line and the web console.

### 12.3.1. Creating an Object Class Using the Command Line

Use the `ldapmodify` utility to create a new object class entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema objectclasses add
examplePerson --oid="2.16.840.1133730.2.123" --desc="Example Person Object Class" --
sup="inetOrgPerson" --kind="AUXILIARY" --must="cn" --may exampleDateOfBirth
examplePreferredOS
```

For further details about object class definitions, see [Section 12.1.2, “Object Classes”](#).

### 12.3.2. Creating an Object Class Using the Web Console

To create an object class using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select **Schema → Objectclasses**.
4. Click **Add ObjectClass**.
5. Select the parent object class.
6. Enter an object class name and, optionally, set a object identifier (OID).
7. Select the object class kind.
8. Enter the required and allowed attributes in the corresponding fields:

**Add ObjectClass - examplePerson**

Objectclass Name	examplePerson
Description	
OID (optional)	2.16.840.1133730.2.123
Parent Objectclass	top
Objectclass Kind	STRUCTURAL
Required Attributes	cn
Allowed Attributes	Type an attribute name...
<input type="button" value="Cancel"/> <input type="button" value="Add"/>	

9. Click **Add**.

## 12.4. UPDATING AN OBJECT CLASS

This section describes how to update an object class using the command line and the web console.

### 12.4.1. Updating an Object Class Using the Command Line

Use the **ldapmodify** utility to update an object class entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema objectclasses edit
examplePerson --oid="2.16.840.1133730.2.123" --desc="Example Person Object Class" --
sup="inetOrgPerson" --kind="AUXILIARY" --must="cn" --may exampleDisplayName exampleAlias
```

For further details about object class definitions, see [Section 12.1.2, "Object Classes"](#).

### 12.4.2. Updating an Object Class Using the Web Console

To update an object class using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Select **Schema** → **Objectclasses**.
4. Click the **Choose Action** button to the right of the object class entry you want to edit.
5. Select **Edit Object Class**.
6. Update the parameters.

**Edit ObjectClass - exampleperson**

Objectclass Name	exampleperson
Description	
OID (optional)	2.16.840.1133730.2.123
Parent Objectclass	top
Objectclass Kind	STRUCTURAL
Required Attributes	cn
Allowed Attributes	Type an attribute name...
<b>Cancel</b> <b>Save</b>	

7. Click **Save**.

## 12.5. REMOVING AN OBJECT CLASS

This section describes how to delete an object class using the command line and the web console.

**WARNING**

Do not delete default schema elements. They are required by Directory Server.

### 12.5.1. Removing an Object Class Using the Command Line

Use the **ldapmodify** utility to delete an object class entry. For example, to delete the **examplePerson** object class:

1. Remove the unwanted attributes from any entries that use them. For details, see [Section 12.8, "Removing an Attribute"](#).
2. Delete the object class entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema objectclasses delete examplePerson
```

For further details about object class definitions, see [Section 12.1.2, "Object Classes"](#).

### 12.5.2. Removing an Object Class Using the Web Console

To remove an object class using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Select **Schema → Objectclasses**.
4. Click the **Choose Action** button next to the object class entry you want to remove.
5. Select **Delete Objectclass**.
6. Click **Yes** to confirm.

## 12.6. CREATING AN ATTRIBUTE

This section describes how to create a attribute using the command line and the web console.

### 12.6.1. Creating an Attribute Using the Command Line

Use the **ldapmodify** utility to create a new attribute. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema attributetypes add dateofbirth --desc="For employee birthdays" --syntax="1.3.6.1.4.1.1466.115.121.1.15" --single-value --x-origin="Example defined"
```

For further details about attribute definitions, see [Section 12.1.3, "Attributes"](#).

## 12.6.2. Creating an Attribute Using the Web Console

To create an attribute using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Select **Schema → Attributes**.
4. Click **Add Attribute**.
5. Fill the parameters.

**Add Attribute - dateOfBirth**

Attribute Name	dateOfBirth
Description	For employee birthdays
OID (optional)	2.16.840.1.1133730.3.1.123
Parent Attribute	Type a parent attribute...
Syntax Name	Directory String
Attribute Usage	userApplications
Multivalued Attribute	<input type="checkbox"/>
Not Modifiable By A User	<input type="checkbox"/>
Alias Names	dob
Equality Matching Rule	Type an matching rule...
Order Matching Rule	Type an matching rule...
Substring Matching Rule	Type an matching rule...

**Cancel** **Add**

6. Click **Add**.

## 12.7. UPDATING AN ATTRIBUTE

This section describes how to update an attribute using the command line and the web console.

### 12.7.1. Updating an Attribute Using the Command Line

Use the **ldapmodify** utility to update an attribute entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema attributetypes edit  
dateofbirth --desc="Employee birthday" --syntax="1.3.6.1.4.1.1466.115.121.1.15" --single-value --x-  
origin="Example defined"
```

For further details about object class definitions, see [Section 12.1.2, “Object Classes”](#).

### 12.7.2. Updating an Attribute Using the Web Console

To update an attribute using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select **Schema → Attributes**.
4. Click the **Choose Action** button next to the attribute you want to edit.
5. Select **Edit Attribute**.
6. Update the parameters.

**Edit Attribute - dateofbirth**

Attribute Name	dateofbirth
Description	For employee birthdays
OID (optional)	2.16.840.1.1133730.3.1.123
Parent Attribute	Type a parent attribute...
Syntax Name	Directory String
Attribute Usage	userApplications
Multivalued Attribute	<input type="checkbox"/>
Not Modifiable By A User	<input type="checkbox"/>
Alias Names	Type an alias name...
Equality Matching Rule	Type an matching rule...
Order Matching Rule	Type an matching rule...
Substring Matching Rule	Type an matching rule...
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

- Click **Save**.

## 12.8. REMOVING AN ATTRIBUTE

This section describes how to delete an attribute using the command line and the web console.

### 12.8.1. Removing an Attribute Using the Command Line

Use the **ldapmodify** utility to delete an attribute. For example:

- Remove the unwanted attributes from any entries which use them.
- Delete the attribute:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema attributetypes
remove dateofbirth
```

For further details about object class definitions, see [Section 12.1.2, “Object Classes”](#).

### 12.8.2. Removing an Attribute Using the Web Console

To remove an attribute using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select **Schema** → **Objectclasses**.
4. Click the **Choose Action** button next to the attribute you want to remove.
5. Select **Delete Attribute**.
6. Click **Yes** to confirm.

## 12.9. CREATING CUSTOM SCHEMA FILES

Schema files are simple LDIF files which define the **cn=schema** entry. Each attribute and object class is added as an attribute to that entry. Here are the requirements for creating a schema file:

- The first line must be **dn: cn=schema**.
- The schema file can include both attributes and object classes, but it can also include only one or the other.
- If both attributes and object classes are defined in the style, all of the attributes must be listed in the file first, then the object classes.
- The object classes can use attributes defined in other schema files.
- The file must be named in the format **[1-9][0-9]text.ldif**.

The file must always begin with two numbers. Numerically, the schema file cannot be loaded before the core configuration schema (which begin with **00** and **01**).

Also, the Directory Server always writes its custom schema to the numerically and alphabetically highest named schema file in the schema directory. It expects this file to be **99user.ldif**. If this file is not **99user.ldif**, the server can experience problems. So, always make sure custom schema files are at least alphabetically lower than **99user.ldif**. The name **99alpha.ldif** is okay; the name **99zzz.ldif** is not.

Practices for creating schema files are described in more detail in the *Deployment Guide*.

Attributes are defined in the schema file as **attributetypes** attributes to the schema, with five components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME name**
- A description, in the form **DESC description**

- The OID for the syntax of the attribute values, discussed in [Section 12.1.3.1, "Directory Server Attribute Syntaxes"](#), in the form **SYNTAX OID**
- Optionally, the source where the attribute is defined

For example:

```
attributetypes: ( 1.2.3.4.5.6.1 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUED X-ORIGIN 'Example defined')
```

Likewise, object classes are defined as values of **objectclasses** attributes, although there is slightly more flexibility in how the object class is defined. The only required configurations are the name and OID for the object class; all other configuration depends on the needs for the object class:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME name**
- A description, in the form **DESC description**
- The superior, or parent, object class for this object class, in the form **SUP object\_class**; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry
- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs (\$)
- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs (\$)

For example:

```
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object
Class' SUP inetOrgPerson AUXILIARY MUST cn MAY (exampleDateOfBirth $ examplePreferredOS) )
```

[Example 12.4, "Example Schema File"](#) shows a simplified schema file.

#### Example 12.4. Example Schema File

```
dn: cn=schema
attributetypes: ( 2.16.840.1133730.1.123 NAME 'dateofbirth' DESC 'For employee birthdays'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'Example defined')
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object
Class' SUP inetOrgPerson AUXILIARY MAY (dateofbirth) )
```

Custom schema files should be added to the Directory Server instance's schema directory, **/etc/dirsrv/slappd-instance/schema**. The schema in these files are not loaded and available to the server unless the server is restarted or a dynamic reload task is run.



## IMPORTANT

If you want to use a standard schema from the `/usr/share/data` directory, copy the schema file to the `/usr/share/dirsrv/schema` directory. If you require that a standard schema is only available to a specific instance, copy the schema file to the `/etc/dirsrv/slappd-instance_name/schema` directory, but use a different file name in the destination directory. Otherwise, Directory Server renames the file during an upgrade and appends the `.bak` suffix.

## 12.10. DYNAMICALLY RELOADING SCHEMA

By default, the schema files used by the Directory Server instance are loaded into the directory when it is started. This means that any new schema files added to the schema directory are not available for use unless the server is restarted. The Directory Server has a task which manually reloads the full schema for the Directory Server instance, including custom files, without requiring a server restart.

You can reload the schema using one of the following methods:

- The **dsconf schema reload** command. See [Section 12.10.1, “Dynamically Reloading the Schema Using the dsconf schema reload Command”](#)
- A **cn=tasks** entry. See [Section 12.10.2, “Dynamically Reloading the Schema Using a cn=tasks Entry”](#)



## NOTE

Directory Server always additionally reloads the all schema files stored in the `/usr/share/dirsrv/schema` directory.

### 12.10.1. Dynamically Reloading the Schema Using the **dsconf schema reload** Command

Use the **dsconf schema reload** command to reload the schema:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema reload
Attempting to add task entry... This will fail if Schema Reload plug-in is not enabled.
Successfully added task entry cn=schema_reload_2018-08-28T09:45:48.027962,cn=schema reload
task,cn=tasks,cn=config
To verify that the schema reload operation was successful, please check the error logs
```

By default, Directory Server reloads the schema files stored in the directory set in the **nsslapd-schemadir** parameter. Optionally, pass the **-d directory** option to the command to reload the schema stored in a different directory.

### 12.10.2. Dynamically Reloading the Schema Using a **cn=tasks** Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate a schema reload operation, create a task in the **cn=schema reload task,cn=tasks,cn=config** entry.

By default, Directory Server reloads the schema files stored in the directory set in the **nsslapd-schemadir** parameter. For example, to reload the schema files stored in the directory set in this parameter:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_schema_reload,cn=schema reload task,cn=tasks,cn=config
objectclass: extensibleObject
cn: cn=example_schema_reload
```

Optionally, specify the **`schemadir`** parameter, to reload the schema stored in a different directory. For example:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_schema_reload,cn=schema reload task,cn=tasks,cn=config
objectclass: extensibleObject
cn: cn=example_schema_reload
schemadir: /example/schema/
```

When the task is completed, the entry is removed from the directory configuration.

For further details about the **`cn=schema reload task,cn=tasks,cn=config`** entry, see the [cn=schema reload task](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

### 12.10.3. Reloading The Schema in a Replication Topology

The schema reload task is a local operation, so schema changes are not replicated in a multi-supplier environment if the schema is added to one supplier but not to the others. To load the new schema files on all of the supplier servers:

1. Stop replication. See [Section 15.9, “Disabling and Re-enabling Replication”](#).
2. Copy the new schema file over and run the schema reload task for every supplier and replica server. See:
  - [Section 12.10.1, “Dynamically Reloading the Schema Using the `dsconf schema reload` Command”](#)
  - [Section 12.10.2, “Dynamically Reloading the Schema Using a `cn=tasks` Entry”](#)
3. Restart replication. See [Section 15.9, “Disabling and Re-enabling Replication”](#).

### 12.10.4. Schema Reload Errors

When the schema reload task runs, the server does not return whether it completed successfully. To verify the schema reload operation was successful, check the error logs. The schema reload has two tasks, first validating the schema file and then loading it.

A success message shows that the validation passed and the task finished.

```
[06/Jan/2009:17:52:04.001214874 -0500] schemareload - Schema reload task starts (schema dir: default) ...
[06/Jan/2009:17:52:04.754335904 -0500] schemareload - Schema validation passed.
[06/Jan/2009:17:52:04.894255328 -0500] schemareload - Schema reload task finished.
```

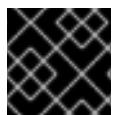
If there is a failure, then the logs show which step failed and why.

```
[..] schemareload - Schema reload task starts (schema dir: /bogus) ...
[..] schema - No schema files were found in the directory /bogus
[..] schema_reload - schema file validation failed
[..] schemareload - Schema validation failed.
```

## 12.11. TURNING SCHEMA CHECKING ON AND OFF

When schema checking is on, the Directory Server ensures three things:

- The object classes and attributes using are defined in the directory schema.
- The attributes required for an object class are contained in the entry.
- Only attributes allowed by the object class are contained in the entry.



### IMPORTANT

Red Hat recommends not to disable the schema checking.

Schema checking is turned on by default in the Directory Server, and the Directory Server should always run with schema checking turned on. The only situation where it may be beneficial to turn schema checking off is to accelerate LDAP import operations. However, there is a risk of importing entries that do not conform to the schema. Consequently, it is impossible to update these entries.

### 12.11.1. Turning Schema Checking On and Off Using the Command Line

To turn schema checking on and off, set the value of the **nsslapd-schemacheck** parameter. For example to disable schema checking:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
schemacheck=off
Successfully replaced "nsslapd-schemacheck"
```

For details about the **nsslapd-schemacheck** parameter, see the description of the parameter in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

### 12.11.2. Turning Schema Checking On and Off Using the Web Console

To enable or disable schema checking using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings**, and select the **Server Settings** entry.
4. Open the **Advanced Settings** tab.
5. To enable schema checking, select the **Enable Schema Checking** check box. To disable the feature, clear the check box.
6. Click **Save**.

## 12.12. USING SYNTAX VALIDATION

With syntax validation, the Directory Server checks that the value of an attribute follows the rules of the syntax given in the definition for that attribute. For example, syntax validation will confirm that a new **telephoneNumber** attribute actually has a valid telephone number for its value.



### IMPORTANT

Red Hat recommends not to disable the syntax validation.

#### 12.12.1. About Syntax Validation

As with schema checking, validation reviews any directory modification and rejects changes that violate the syntax. Additional settings can be optionally configured so that syntax validation can log warning messages about syntax violations and then either reject the modification or allow the modification process to succeed.

This feature validates all attribute syntaxes, with the exception of binary syntaxes (which cannot be verified) and non-standard syntaxes, which do not have a defined required format. The syntaxes are validated against [RFC 4514](#).

#### 12.12.2. Syntax Validation and Other Directory Server Operations

Syntax validation is mainly relevant for standard LDAP operations like creating entries (add) or editing attributes (modify). Validating attribute syntax can impact other Directory Server operations, however.

##### Database Encryption

For normal LDAP operations, an attribute is encrypted just before the value is written to the database. This means that encryption occurs *after* the attribute syntax is validated.

Encrypted databases (as described in [Chapter 10, Configuring Attribute Encryption](#)) can be exported and imported. Normally, it is strongly recommended that these export and import operations are done with the **-E** flag with **db2ldif** and **ldif2db**, which allows syntax validation to occur just fine for the import operation. However, if the encrypted database is exported without using the **-E** flag (which is not supported), then an LDIF with encrypted values is created. When this LDIF is then imported, the encrypted attributes cannot be validated, a warning is logged, and attribute validation is skipped in the imported entry.

##### Synchronization

There may be differences in the allowed or enforced syntaxes for attributes in Windows Active Directory entries and Red Hat Directory Server entries. In that case, the Active Directory values could not be properly synchronized over because syntax validation enforces the RFC standards in the Directory Server entries.

##### Replication

If the Directory Server 11 instance is a supplier which replicates its changes to a consumer, then there is no issue with using syntax validation. However, if the supplier in replication is an older version of Directory Server or has syntax validation disabled, then syntax validation should not be used on the consumer because the Directory Server 11 consumer may reject attribute values that the supplier allows.

#### 12.12.2.1. Turning Syntax Validation On and Off Using the Command Line

To turn syntax validation on and off, set the value of the ***nsslapd-syntaxcheck*** parameter. For example to disable syntax validation:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
syntaxcheck=off
Successfully replaced "nsslapd-syntaxcheck"
```

For details about the ***nsslapd-syntaxcheck*** parameter, see the description of the parameter in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

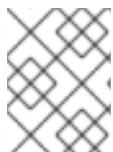
### 12.12.2.2. Turning Syntax Validation On and Off Using the Web Console

To enable or disable syntax validation using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings**, and select the **Server Settings** entry.
4. Open the **Advanced Settings** tab.
5. To enable attribute syntax checking, select the **Enable Attribute Syntax Checking** check box. To disable the feature, clear the check box.
6. Click **Save**.

### 12.12.3. Enabling or Disabling Strict Syntax Validation for DNs

When syntax validation is enabled, distinguished names (DN) are validated as described in section 3 of [RFC 4514](#). DN syntax validation is enabled separately because the strictness of later standards can invalidate older DNs that have a different syntax, and therefore directory trees.



#### NOTE

If strict DN validation is enabled and a DN value does not conform to the required syntax, then the operation fails with the LDAP result code **34, INVALID\_DN\_SYNTAX**.

### 12.12.3.1. Enabling or Disabling Strict Syntax Validation for DNs Using the Command Line

To turn strict syntax validation for DNs on and off, set the value of the ***nsslapd-dn-validate-strict*** parameter. For example to disable strict syntax validation for DNs::

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-dn-validate-
strict=off
Successfully replaced "nsslapd-dn-validate-strict"
```

For details about the ***nsslapd-syntaxcheck*** parameter, see the description of the parameter in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

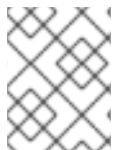
### 12.12.3.2. Enabling or Disabling Strict Syntax Validation for DNs Using the Web Console

To enable or disable strict syntax validation for DNs using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings**, and select the **Server Settings** entry.
4. Open the **Advanced Settings** tab.
5. Select or unselect the **Strict DN Syntax Validation** option depending on whether you want to enable or disable the feature.
6. Click **Save**.

#### 12.12.4. Enabling Syntax Validation Logging

By default, syntax validation rejects any add or modify operations where an attribute value violates the required syntax. However, the violation itself is not recorded to the errors log by default. The **nsslapd-syntaxlogging** attribute enables error logging for any syntax violations.



##### NOTE

Syntax violations discovered by the syntax validation script and task are logged in the Directory Server error log.

If both the **nsslapd-syntaxlogging** and **nsslapd-syntaxcheck** parameter are enabled, any invalid attribute modification is rejected and the message written to the log. If **nsslapd-syntaxlogging** is enabled but **nsslapd-syntaxcheck** is disabled, then the operation is allowed to succeed, but the warning message is still written to the error log.

##### 12.12.4.1. Enabling Syntax Validation Logging Using the Command Line

To enable syntax validation logging, set the value of the **nsslapd-syntaxlogging** parameter to **on**.

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-syntaxlogging=on
Successfully replaced "nsslapd-syntaxlogging"
```

For details about the **nsslapd-syntaxlogging** parameter, see the description of the parameter in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

##### 12.12.4.2. Enabling Syntax Validation Logging Using the Web Console

To enable validation logging using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings**, and select the **Server Settings** entry.
4. Open the **Advanced Settings** tab.

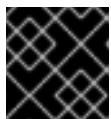
5. Select the **Enable Attribute Syntax Logging** option.

6. Click **Save**.

### 12.12.5. Validating the Syntax of Existing Attribute Values

In certain situations, you might want to manually validate the syntax of existing values. For example:

- If syntax validation is disabled in the ***nsslapd-syntaxcheck*** parameter. For details, see [Section 12.12.2, “Syntax Validation and Other Directory Server Operations”](#).



#### IMPORTANT

Red Hat recommends not disabling syntax validation.

- If you migrate data from a server without or disabled syntax validation.

Directory Server logs results of syntax validation tasks to the **/var/log/dirsrv/slapd-*instance\_name*/errors** file. For example:

- If all verified values are valid:

```
[28/Jun/2017:12:52:43.669867966 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Starting (base: "dc=example,dc=com", filter: "(objectclass=*)")
...
[28/Jun/2017:12:52:43.696850129 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Complete. Found 0 invalid entries.
```

- If invalid entries were found:

```
[28/Jun/2017:12:54:05.736087520 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Starting (base: "dc=example,dc=com", filter: "(objectclass=*)")
...
[28/Jun/2017:12:54:05.754195607 +0200] - ERR - syntax-plugin -
syntax_validate_task_callback - Entry "cn=user,ou=People,dc=example,dc=com" violates
syntax.
description: value #0 invalid per syntax
[28/Jun/2017:12:54:05.759905671 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Complete. Found 1 invalid entries.
```



#### NOTE

The syntax validation task identifies only syntax violations. You must fix incorrect values manually.

### 12.12.5.1. Creating a Syntax Validation Task Using the **dsconf schema validate-syntax** Command

Use the **dsconf schema validate-syntax** command to create a syntax validation task. For example, to create a task that validates the syntax of all values in the **ou=People,dc=example,dc=com** sub-tree which match the **(objectclass=inetorgperson)** filter, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema validate-syntax -f  
'(objectclass/inetorgperson)' ou=People,dc=example,dc=com
```

#### 12.12.5.2. Creating a Syntax Validation Task Using a cn=tasks Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate a syntax validation operation, create a task in the **cn=syntax validate,cn=tasks,cn=config** entry.

For example, to create a task that validates the syntax of all values in the **ou=People,dc=example,dc=com** sub-tree which match the **(objectclass=inetorgperson)** filter:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
  
dn: cn=example_syntax_validate,cn=syntax validate,cn=tasks,cn=config  
objectclass: extensibleObject  
cn: cn=example_syntax_validate  
basedn: ou=People,dc=example,dc=com  
filter: (objectclass=inetorgperson)
```

When the task is completed, the entry is removed from the directory configuration.

For further details, about the **cn=syntax validate,cn=tasks,cn=config** entry, see the [cn=schema reload task](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

# CHAPTER 13. MANAGING INDEXES

Indexing makes searching for and retrieving information easier by classifying and organizing attributes or values. This chapter describes the searching algorithm itself, placing indexing mechanisms in context, and then describes how to create, delete, and manage indexes.

## 13.1. ABOUT INDEXES

This section provides an overview of indexing in Directory Server. It contains the following topics:

- [Section 13.1.1, "About Index Types"](#)
- [Section 13.1.2, "About Default and Database Indexes"](#)
- [Section 13.1.3, "Overview of the Searching Algorithm"](#)
- [Section 13.1.5, "Balancing the Benefits of Indexing"](#)

### 13.1.1. About Index Types

Indexes are stored in files in the directory's databases. The names of the files are based on the indexed attribute, not the type of index contained in the file. Each index file may contain multiple types of indexes if multiple indexes are maintained for the specific attribute. For example, all indexes maintained for the common name attribute are contained in the **cn.db** file.

Directory Server supports the following types of index:

- *Presence index (pres)* contains a list of the entries that contain a particular attribute, which is very useful for searches. For example, it makes it easy to examine any entries that contain access control information. Generating an **aci.db** file that includes a presence index efficiently performs the search for **ACI=\*** to generate the access control list for the server.
- *Equality index (eq)* improves searches for entries containing a specific attribute value. For example, an equality index on the **cn** attribute allows a user to perform the search for **cn=Babs Jensen** far more efficiently.
- *Approximate index (approx)* is used for efficient approximate or *sounds-like* searches. For example, an entry may include the attribute value **cn=Firstname M Lastname**. An approximate search would return this value for searches against **cn~=Firstname Lastname**, **cn~=Firstname**, or **cn~=Lastname**. Similarly, a search against **I~=San Fransisco** (note the misspelling) would return entries including **I=San Francisco**.
- *Substring index (sub)* is a costly index to maintain, but it allows efficient searching against substrings within entries. Substring indexes are limited to a minimum of three characters for each entry.

For example, searches of the form **cn=\*&derson**, match the common names containing strings such as **Bill Anderson**, **Jill Henderson**, or **Steve Sanderson**. Similarly, the search for **telephoneNumber= \*555\*** returns all the entries in the directory with telephone numbers that contain **555**.

- *International index* speeds up searches for information in international directories. The process for creating an international index is similar to the process for creating regular indexes, except that it applies a *matching rule* by associating an *object identifier* (OID) with the attributes to be indexed.

The supported locales and their associated OIDs are listed in [Appendix D, Internationalization](#). If there is a need to configure the Directory Server to accept additional matching rules, contact Red Hat Consulting.

### 13.1.2. About Default and Database Indexes

Directory Server contains a set of default indexes. When you create a new database, Directory Server copies these default indexes from **`cn=default indexes,cn=config,cn=ldbmdatabase,cn=plugins,cn=config`** to the new database. Then the database only uses the copy of these indexes, which are stored in **`cn=index,cn=database_name,cn=ldbmdatabase,cn=plugins,cn=config`**.



#### NOTE

Directory Server does not replicate settings in the `cn=config` entry. Therefore, you can configure indexes differently on servers that are part of a replication topology. For example, in an environment with cascading replication, you do not need to create custom indexes on a hub, if clients do not read data from the hub.

To display the Directory Server default indexes:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com \
-b "cn=default indexes,cn=config,cn=ldbmdatabase,cn=plugins,cn=config" \
'(objectClass=nsindex)'
```



#### NOTE

If you update the default index settings stored in **`cn=default indexes,cn=config,cn=ldbmdatabase,cn=plugins,cn=config`**, the changes are not applied to the individual databases in **`cn=index,cn=database_name,cn=ldbmdatabase,cn=plugins,cn=config`**.

To display the indexes of an individual database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend index list database_name
```

### 13.1.3. Overview of the Searching Algorithm

Indexes are used to speed up searches. To understand how the directory uses indexes, it helps to understand the searching algorithm. Each index contains a list of attributes (such as the **`cn`**, common name, attribute) and a list of IDs of the entries which contain the indexed attribute value:

1. An LDAP client application sends a search request to the directory.
2. The directory examines the incoming request to make sure that the specified base DN matches a suffix contained by one or more of its databases or database links.
  - o If they do match, the directory processes the request.
  - o If they do not match, the directory returns an error to the client indicating that the suffix does not match. If a referral has been specified in the **`nsslapd-referral`** attribute under **`cn=config`**, the directory also returns the LDAP URL where the client can attempt to pursue the request.

- The Directory Server examines the search filter to see what indexes apply, and it attempts to load the list of entry IDs from each index that satisfies the filter. The ID lists are combined based on whether the filter used AND or OR joins.

Each filter component is handled independently and returns an ID list.

- If the list of entry IDs is larger than the configured ID list scan limit or if there is no index defined for the attribute, then Directory Server sets the results for this **filtercomponent** to **allids**. If, after applying the logical operations to the results of the individual search components the list is still **ALLIDS** it searches every entry in the database. This is an *unindexed* search.

- The Directory Server reads every entry from the **id2entry.db** database or the entry cache for every entry ID in the ID list (or from the entire database for an unindexed search). The server then checks the entries to see if they match the search filter. Each match is returned as it is found.

The server continues through the list of IDs until it has searched all candidate entries or until it hits one of the configured resource limits. (Resource limits are listed in [Section 14.4.3, “Setting User and Global Resource Limits Using the Command Line”](#).)



#### NOTE

It's possible to set separate resource limits for searches using the simple paged results control. For example, administrators can set high or unlimited size and look-through limits with paged searches, but use the lower default limits for non-paged searches.

### 13.1.4. Approximate Searches

In addition, the directory uses a variation of the metaphone phonetic algorithm to perform searches on an approximate index. Each value is treated as a sequence of words, and a phonetic code is generated for each word.



#### NOTE

The metaphone phonetic algorithm in Directory Server supports only US-ASCII letters. Therefore, use approximate indexing only with English values.

Values entered on an approximate search are similarly translated into a sequence of phonetic codes. An entry is considered to match a query if both of the following are true:

- All of the query string codes match the codes generated in the entry string.
- All of the query string codes are in the same order as the entry string codes.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
Alice B Sarette (ALS B SRT)	Alice Sarette (ALS SRT)	Matches. Codes are specified in the correct order.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
	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 (BRO SRT)	No match. The code BRO does not exist in the original name.
	Sarette, Alice (SRT ALS)	No match. The codes are not specified in the correct order.

### 13.1.5. Balancing the Benefits of Indexing

Before creating new indexes, balance the benefits of maintaining indexes against the costs.

- Approximate indexes are not efficient for attributes commonly containing numbers, such as telephone numbers.
- Substring indexes do not work for binary attributes.
- Equality indexes should be avoided if the value is big (such as attributes intended to contain photographs or passwords containing encrypted data).
- Maintaining indexes for attributes not commonly used in a search increases overhead without improving global searching performance.
- Attributes that are not indexed can still be specified in search requests, although the search performance may be degraded significantly, depending on the type of search.
- The more indexes you maintain, the more disk space you require.

Indexes can become very time-consuming. For example:

1. The Directory Server receives an add or modify operation.
2. The Directory Server examines the indexing attributes to determine whether an index is maintained for the attribute values.
3. If the created attribute values are indexed, then Directory Server adds or deletes the new attribute values from the index.
4. The actual attribute values are created in the entry.

For example, the Directory Server adds the entry:

```
dn: cn=John Doe,ou=People,dc=example,dc=com
objectclass: top
objectClass: person
```

```

objectClass: orgperson
objectClass: inetorgperson
cn: John Doe
cn: John
sn: Doe
ou: Manufacturing
ou: people
telephoneNumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.

```

The Directory Server maintains the following indexes:

- Equality, approximate, and substring indexes for ***cn*** (common name) and ***sn*** (surname) attributes.
- Equality and substring indexes for the telephone number attribute.
- Substring indexes for the description attribute.

When adding that entry to the directory, the Directory Server must perform these steps:

1. Create the ***cn*** equality index entry for **John** and **John Doe**.
2. Create the appropriate ***cn*** approximate index entries for **John** and **John Doe**.
3. Create the appropriate ***cn*** substring index entries for **John** and **John Doe**.
4. Create the ***sn*** equality index entry for **Doe**.
5. Create the appropriate ***sn*** approximate index entry for **Doe**.
6. Create the appropriate ***sn*** substring index entries for **Doe**.
7. Create the telephone number equality index entry for **408 555 8834**.
8. Create the appropriate telephone number substring index entries for **408 555 8834**.
9. Create the appropriate description substring index entries for **Manufacturing lead for the Z238 line of widgets**. A large number of substring entries are generated for this string.

As this example shows, the number of actions required to create and maintain databases for a large directory can be resource-intensive.

### 13.1.6. Indexing Limitations

You cannot index virtual attributes, such as ***nsrole*** and ***cos\_attribute***. Virtual attributes contain computed values. If you index these attributes, Directory Server can return an invalid set of entries to direct and internal searches.

## 13.2. CREATING STANDARD INDEXES

This section describes how to create presence, equality, approximate, substring, and international indexes for specific attributes using the command line and the web console.

**NOTE**

When you create a new index type, Directory Server uses this default index as a template for each new database that will be created in future. If you update the default index, the updated settings are not applied to existing databases. To apply a new index to an existing database, use the **dsctl db2index** command or a **cn=index,cn=tasks** task, as described in [Section 13.3, “Creating New Indexes to Existing Databases”](#).

- [Section 13.2.2, “Creating Indexes Using the Web Console”](#)
- [Section 13.2.1, “Creating Indexes Using the Command Line”](#)

### 13.2.1. Creating Indexes Using the Command Line

**NOTE**

You cannot create new system indexes because system indexes are hard-coded in Directory Server.

Use **ldapmodify** to add the new index attributes to your directory.

- To create a new index that will become one of the default indexes, add the new index attributes to the **cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config** entry.
- To create a new index for a particular database, add it to the **cn=index,cn=database\_name,cn=ldbm database,cn=plugins,cn=config** entry, where **cn=database\_name** corresponds to the name of the database.

**NOTE**

Avoid creating entries under **cn=config** in the **dse.ldif** file. The **cn=config** entry in the **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will probably suffer. Although we recommend you do not store simple user entries under **cn=config** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **cn=config** since this centralizes configuration information.

For information on the LDIF update statements required to add entries, see [Section 3.4, “Updating a Directory Entry”](#).

For example, to create presence, equality, and substring indexes for the **sn** (surname) attribute in the **Example1** database:

1. Run **ldapmodify** and add the LDIF entry for the new indexes:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectClass:top
objectClass:nsIndex
```

```

cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1

```

The **cn** attribute contains the name of the attribute to index, in this example the **sn** attribute. The entry is a member of the **nsIndex** object class. The **nsSystemIndex** attribute is **false**, indicating that the index is not essential to Directory Server operations. The multi-valued **nsIndexType** attribute specifies the presence (**pres**), equality (**eq**) and substring (**sub**) indexes. Each keyword has to be entered on a separate line. The **nsMatchingRule** attribute in the example specifies the OID of the Bulgarian collation order; the matching rule can indicate any possible value match, such as languages or other formats like date or integer.

You can use the keyword **none** in the **nsIndexType** attribute to specify that no indexes are to be maintained for the attribute. This example temporarily disables the **sn** indexes on the **Example1** database by changing the **nsIndexType** to **none**:

```

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:none

```

For a complete list of matching rules and their OIDs, see [Section 14.2.4, “Using Matching Rules”](#), and for the index configuration attributes, see the *Red Hat Directory Server Configuration, Command, and File Reference*.



## NOTE

Always use the attribute's primary name (not the attribute's alias) when creating indexes. The primary name of the attribute is the first name listed for the attribute in the schema; for example, **uid** for the user ID attribute.

### 13.2.2. Creating Indexes Using the Web Console

To create presence, equality, approximate, substring, or international indexes:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** menu.
4. Select the suffix entry.
5. Open the **Indexes** tab.
6. Click the **Add Index** button.
7. Select the attribute to index, the type of index, and optionally a matching rule.

## Add Database Index



Select An Attribute

buildingname



### Index Types

- Equality Indexing
- Presence Indexing
- Substring Indexing
- Approximate Indexing

### Matching Rules

Type a matching rule name...



Index attribute after creation

[Cancel](#)

[Create Index](#)

8. Click **Create Index**.

## 13.3. CREATING NEW INDEXES TO EXISTING DATABASES

Learn how to initiate indexing operations on Directory Server. You must create indexes manually because Directory Server does not automatically index databases.



### IMPORTANT

Before you regenerate the index, searches proceed but can return incorrect or inconsistent results.

### 13.3.1. Creating an Index While the Instance is Running

#### 13.3.1.1. Creating an Index Using the `dsconf backend index reindex` Command

To recreate the index of a database while the instance is running:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend index reindex
database_name
```

#### 13.3.1.2. Creating an Index Using a `cn=tasks` Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate an index operation, create a task in the **cn=index,cn=tasks,cn=config** entry.

Use the **ldapadd** utility to add a new index task. For example, to add a task that creates the presence index for the **cn** attribute in the **userRoot** database:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=example_presence_index,cn=index,cn=tasks,cn=config
objectclass: top
objectclass: extensibleObject
cn: example presence index
nsInstance: userRoot
nsIndexAttribute: "cn:pres"
```

When the task is completed, the entry is removed from the directory configuration.

For further details, about the **cn=index,cn=tasks,cn=config** entry, see the [cn=index](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

### 13.3.2. Creating an Index While the Instance Offline

After creating an indexing entry or adding additional index types to an existing indexing entry, use the **dsconf db2index** command:

1. Shut down the instance:

```
# dsctl instance_name stop
```

2. Recreate the index:

```
# dsctl instance_name db2index userRoot
[13/Aug/2019:15:25:37.277426483 +0200] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[13/Aug/2019:15:25:37.289257996 +0200] - INFO - check_and_set_import_cache -
pagesize: 4096, available bytes 1704378368, process usage 22212608
[13/Aug/2019:15:25:37.291738104 +0200] - INFO - check_and_set_import_cache - Import
allocates 665772KB import cache.
...
db2index successful
```

3. Start the instance:

```
# dsctl instance_name start
```

## 13.4. CREATING BROWSING INDEXES

A *virtual list view (VLV) index* is a way of creating a partial list, defined by the VLV search parameters, 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.

A browsing index is a type of VLV index that organizes the entries listed into alphabetical order, making it easier to find entries.

VLV indexes are not applied to attributes, like standard indexes are, but they are dynamically generated based on attributes set in entries and the location of those entries in the directory tree. VLV indexes,

unlike standard indexes, are special entries in the database rather than configuration settings for the database.



## NOTE

VLV indexes are similar to simple paged results, which can be returned with some external LDAP clients. Simple paged results are calculated per search, while VLV indexes are a permanent list, so VLV indexes are overall faster for searches, but do require some overhead for the server to maintain.

Simple paged results and VLV indexes *cannot* be used on the same search.

For more information, see [Section 14.6.4, “Using Simple Paged Results”](#).

### 13.4.1. Creating Browsing Indexes from the Command Line

Creating a browsing index or virtual list view (VLV) index from the command line has these steps:

1. Using **ldapmodify** to add new browsing index entries or edit existing browsing index entries. See [Section 13.4.1.1, “Adding a Browsing Index Entry”](#).
2. Running the **dsconf backend vlv-index reindex** command to generate the new set of browsing indexes to be maintained by the server. See [Section 13.4.1.2, “Recreating the VLV Index”](#). Alternatively, launch an appropriate task under **cn=tasks,cn=config** ([Section 13.4.1.3, “Creating a Browsing Index Using a cn=tasks Entry”](#)).
3. Ensuring that access control on VLV index information is set appropriately. See [Section 13.4.2, “Setting Access Control for VLV Information”](#).

#### 13.4.1.1. Adding a Browsing Index Entry

The type of browsing index entry to create depends on the type of **ldapsearch** attribute sorting to accelerate. It is important to take the following into account:

- The scope of the search (base, one, sub)
- The base of the search (the entry to use as a starting point for the search)
- The attributes to sort
- The filter of the search

For more information on specifying filters for searches, see [Chapter 14, Finding Directory Entries](#).

- The LDBM database to which the entry that forms the base of the search belongs. You can only create browsing indexes in LDBM databases.

For example, create a browsing index to accelerate an **ldapsearch** on the entry **ou=People,dc=example,dc=com** held in the **userRoot** database with the following attributes:

- The search base is **ou=People,dc=example,dc=com**
- The search filter is **(ou=accounting)**
- The scope is **one**

- The sorting order for the returned attributes is ***cn*, *givenname*, *o*, *ou*, and *sn***

  - Run **ldapmodify** and add an entry which specifies the base, scope, and filter of the browsing index:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm
database,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People,dc=example,dc=com
vlvScope: 1
vlvFilter: (ou=accounting)
```

- The ***cn*** contains the browsing index identifier, which specifies the entry on which to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry.
  - The **vlvbase** attribute value specifies the entry on which you want to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry (the browsing index identifier).
  - The **vlvScope** attribute is **1**, indicating that the scope for the search you want to accelerate is **1**. A search scope of **1** means that only the immediate children of the entry specified in the ***cn*** attribute, and not the entry itself, will be searched.
  - The **vlvFilter** specifies the filter to be used for the search; in this example, **(ou=accounting)**.
- Add the second entry, to specify the sorting order for the returned attributes:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

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 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.

#### 13.4.1.2. Recreating the VLV Index

After creating the two browsing indexing entries or added additional attribute types to an existing indexing browsing entries, run the **dsconf backend vlv-index reindex** command to generate the new set of browsing indexes to be maintained by the Directory Server. After running the command, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

1. Stop the server.

```
# dsctl instance_name stop
```

2. Start the reindex process:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend vlv-index reindex --parent-name="by MCC ou=People dc=example dc=com" userRoot
```

3. Start the server.

```
# dsctl instance_name start
```

#### 13.4.1.3. Creating a Browsing Index Using a **cn=tasks** Entry

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries the server uses to manage tasks. To initiate an index operation, create a task in the **cn=index,cn=tasks,cn=config** entry.

Use the **ldapadd** utility to add a new browsing index task. For example:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example_VLV_index,cn=index,cn=tasks,cn=config
objectclass: top
objectclass: extensibleObject
cn: example_VLV_index
nsIndexVLVAttribute: "by MCC ou=people,dc=example,dc=com"
```

Set the **nsIndexVLVAttribute** attribute to the relative distinguished name RDN of the subentry of the VLV entry definition. The previous example uses the RDN created in [Section 13.4.1.1, “Adding a Browsing Index Entry”](#).

The preceding task initiates an operation on Directory Server to creates the index. When the task completes, the **cn=tasks,cn=config** entry is removed.

For further information, see the **cn=index** section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

#### 13.4.2. Setting Access Control for VLV Information

The default access control instruction (ACI) allows only authenticated users to use the VLV index information. If you additionally require to allow non-authenticated users to use the VLV index information, update the **aci** attribute to set the **userdn** parameter to **ldap://anyone**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

```

dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: (targetattr != "aci")(version 3.0; acl "VLV Request Control";
    allow( read, search, compare, proxy ) userdn = "ldap://anyone" ;)

```

## 13.5. CHANGING THE INDEX SORT ORDER

By default, indexes are sorted alphabetically, in descending ASCII order. This is true for every attribute, even attributes which may have numeric attribute values like Integer or TelephoneNumber. It is possible to change the sort method by changing the matching rule set for the attribute.

### 13.5.1. Changing the Sort Order Using the Command Line

To change the sort order using the command line, change the ***nsMatchingRule*** for the attribute index. For example:

```

# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
changetype:modify
replace:nsMatchingRule
nsMatchingRule: integerOrderingMatch

```

## 13.6. CHANGING THE WIDTH FOR INDEXED SUBSTRING SEARCHES

By default, for a search to be indexed, the search string must be at least three characters long, without counting any wildcard characters. For example, the string **abc** would be an indexed search while **ab\*** would not be. Indexed searches are significantly faster than unindexed searches, so changing the minimum length of the search key is helpful to increase the number of indexed searches.

To improve search performance, particularly for sites with many wildcard searches, the search string length for indexed searches can be changed. Directory Server has three attributes which allow you to change the minimum number of characters required for an indexed search:

- The ***nsSubStrBegin*** attribute sets the required number of characters for an indexed search for the beginning of a search string, before the wildcard.

```
abc*
```

- The ***nsSubStrMiddle*** attribute sets the required number of characters for an indexed search where a wildcard is used in the middle of a search string. For example:

```
*abc*
```

- The ***nsSubStrEnd*** attribute sets the required number of characters for an indexed search for the end of a search string, after the wildcard. For example:

```
*xyz
```

The default substring search length for the string triplet (before, middle, and end) is 3, 3, and 3, meaning every search requires a minimum of three characters, in every wildcard position.

For any attribute index to have alternate string lengths, add the **extensibleObject** object class to the entry and then set the substring search lengths.

- Set the new key length for the specific attribute index. This requires adding the **extensibleObject** object class and then adding the **nsSubStrBegin**, **nsSubStrEnd**, or **nsSubStrMiddle** attributes as appropriate. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: attribute_name,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
add: objectclass
objectclass: extensibleObject
-
add: nsSubStrBegin
nsSubStrBegin: 2
-
add: nsSubStrMiddle
nsSubStrMiddle: 2
-
add: nsSubStrEnd
nsSubStrEnd: 2
```

## 13.7. DELETING INDEXES

This section describes how to remove attributes and index types from the index.

### 13.7.1. Deleting an Attribute from the Default Index Entry

When using the default settings of Directory Server, several attributes listed in the default index entry, such as **sn**, are indexed. The following attributes are part of the default index:

**Table 13.1. Default Index Attributes**

aci	cn	entryusn
givenName	mail	mailAlternateAddress
mailHost	member	memberOf
nsUniqueld	ntUniqueld	ntUserDomainId
numsubordinates	objectclass	owner
parentid	seeAlso	sn
telephoneNumber	uid	uniquemember

**WARNING**

Removing system indexes can significantly affect the Directory Server performance.

For example, to remove the ***sn*** attribute from the default index:

1. Remove the attribute from the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** entry:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
cn=sn,cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config
```

If you do not remove the attribute from this entry, the index for the ***sn*** attribute is automatically recreated and corrupted after the server is restarted.

2. Remove the ***cn=attribute\_name,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config*** entry. For details, see [Section 13.7.2, “Removing an Attribute from the Index”](#)

## 13.7.2. Removing an Attribute from the Index

In certain situations you want to remove an attribute from the index. This section describe the procedure using the command line and using the web console.

### 13.7.2.1. Removing an Attribute from the Index Using the Command Line

To remove an attribute from the index:

1. If the attribute to remove is listed in the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** default index entry, remove it from this entry first. For details, see [Section 13.7.1, “Deleting an Attribute from the Default Index Entry”](#).
2. Remove the attribute from the index. For example:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
```

After deleting the entry, Directory Server no longer maintains the index for the attribute.

3. Recreate the attribute index. See [Section 13.3, “Creating New Indexes to Existing Databases”](#).

### 13.7.2.2. Removing an Attribute from the Index Using the Web Console

To remove an attribute from the index:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.

3. Open the **Database** menu.
4. Select the suffix entry.
5. Open the **Indexes** tab.
6. Click the **Actions** button next to the attribute for which you want to remove the index, and select **Delete Index**.
7. Click **Yes** to confirm.

### 13.7.3. Deleting Index Types Using the Command Line

For example, to remove the **sub** index type of the **sn** attribute from the index:

1. Remove the index type:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
delete: nsIndexType
nsIndexType: sub
```

After deleting the index entry, Directory Server no longer maintains the substring index for the attribute.

2. Recreate the attribute index. See [Section 13.3, “Creating New Indexes to Existing Databases”](#).

### 13.7.4. Removing Browsing Indexes

This section describes how to remove browsing entries from a database.

#### 13.7.4.1. Removing Browsing Indexes Using the Command Line

The entries for an alphabetical browsing index and virtual list view (VLV) are the same. This section describes the steps involved in removing browsing indexes.

To remove a browsing index or virtual list view index using the command line:

1. Remove the browsing index entries from the **cn=index,cn=database\_name,cn=ldbm database,cn=plugins,cn=config** entry. For example:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x "cn=MCC
ou=People dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config"
"cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People dc=example
dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config"
```

After deleting the two browsing index entries, Directory Server no longer maintains these indexed.

2. Recreate the attribute index. See [Section 13.3, “Creating New Indexes to Existing Databases”](#).

# CHAPTER 14. FINDING DIRECTORY ENTRIES

Entries in the directory can be searched for and found using any LDAP client. Most clients provide some form of search interface so that the directory can be searched easily and entry information can be easily retrieved.

## 14.1. USING LDAPSEARCH

The **ldapsearch** command-line utility can locate and retrieve directory entries. This utility opens a connection to the specified server using the specified identity and credentials and locates entries based on a specified search filter. The search scope can include a single entry (**-s base**), an entry's immediate subentries (**-s one**), or an entire tree or subtree (**-s sub**).



### NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=bjensen,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless **dc:example** has explicitly been added as an attribute in that entry.

Search results are returned in LDIF format. LDIF is defined in [RFC 2849](#) and is described in detail in [Appendix B, "LDAP Data Interchange Format"](#).

This section contains information about the following topics:

- [Section 14.1.1, "ldapsearch Command-Line Format"](#)
- [Section 14.1.2, "Commonly Used ldapsearch Options"](#)
- [Section 14.1.3, "Using Special Characters"](#)

### 14.1.1. ldapsearch Command-Line Format

The **ldapsearch** command must use the following format:

```
# ldapsearch [-x | -Y mechanism] [options] [search_filter] [list_of_attributes]
```

- Either **-x** (to use simple binds) or **-Y** (to set the SASL mechanism) must be used to configure the type of connection.
- *options* is a series of command-line options. These must be specified before the search filter, if any are used.
- *search\_filter* is an LDAP search filter as described in [Section 14.2, "LDAP Search Filters"](#). Do not specify a separate search filter if search filters are specified in a file using the **-f** option.
- *list\_of\_attributes* is a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see [Section 14.3.6, "Displaying Subsets of Attributes"](#). If a list of attributes is not specified, the search returns values for all attributes permitted by the access control set in the directory, with the exception of operational attributes.

For operational attributes to be returned as a result of a search operation, they must be explicitly specified in the search command. To return all operational attributes of an object specify **-+**. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (\*) in the list of attributes in the **ldapsearch** command.

To retrieve only a list of matching DNs, use the special attribute **1.1**. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com \
-b "dc=example,dc=com" -x "(objectclass=inetorgperson)" 1.1
```

### 14.1.2. Commonly Used **ldapsearch** Options

The following table lists the most commonly used **ldapsearch** command-line options. If a specified value contains a space ( ), the value should be surrounded by single or double quotation marks, such as **-b "cn=My Special Group,ou=groups,dc=example,dc=com"**.



#### IMPORTANT

The **ldapsearch** utility from OpenLDAP uses SASL connections by default. To perform a simple bind or to use TLS, use the **-x** argument to disable SASL and allow other connection methods.

Option	Description
<b>-b</b>	<p>Specifies the starting point for the search. The value specified here must be a distinguished name that currently exists in the database. This is optional if the <b>LDAP_BASEDN</b> environment variable has been set to a base DN. The value specified in this option should be provided in single or double quotation marks. For example:</p> <pre>-b "cn=user,ou=Product Development,dc=example,dc=com"</pre> <p>To search the root DSE entry, specify an empty string here, such as <b>-b ""</b>.</p>
<b>-D</b>	<p>Specifies the distinguished name with which to authenticate to the server. This is optional if anonymous access is supported by the server. If specified, this value must be a DN recognized by the Directory Server, and it must also have the authority to search for the entries. For example, <b>-D "uid=user_name,dc=example,dc=com"</b>.</p>

Option	Description
-H	<p>Specifies an LDAP URL to use to connect to the server. For a traditional LDAP URL, this has the following format:</p> <p><b>ldap[s]://hostname[:port]</b></p> <p>The <i>port</i> is optional; it will use the default LDAP port of 389 or LDAPS port of 636 if the port is not given.</p> <p>This can also use an LDAPI URL, with each element separated by the HTML hex code <b>%2F</b>, rather than a forward slash (/):</p> <p><b>ldapi://%2Ffull%2Fpath%2Fto%2Fslapd-example.socket</b></p> <p>For LDAPI, specify the full path and filename of the LDAPI socket the server is listening to. Since this value is interpreted as an LDAP URL, the forward slash characters (/) in the path and filename must be escaped encoded as the URL escape value <b>%2F</b>.</p> <p>The <b>-H</b> option is used instead of <b>-h</b> and <b>-p</b>.</p>
-h	<p>Specifies the host name or IP address of the machine on which the Directory Server is installed. For example, <b>-h server.example.com</b>. If a host is not specified, <b>Idapsearch</b> uses the localhost.</p> <p> <b>NOTE</b></p> <p>Directory Server supports both IPv4 and IPv6 addresses.</p>
-l	<p>Specifies the maximum number of seconds to wait for a search request to complete. For example, <b>-l 300</b>. The default value for the <b>nsslapd-timelimit</b> attribute is <b>3600</b> seconds. Regardless of the value specified, <b>Idapsearch</b> will never wait longer than is allowed by the server's <b>nsslapd-timelimit</b> attribute.</p>
-p	<p>Specifies the TCP port number that the Directory Server uses. For example, <b>-p 1049</b>. The default is <b>389</b>. If <b>-h</b> is specified, <b>-p</b> must also be specified, even if it gives the default value.</p>
-s scope	<p>Specifies the scope of the search. The scope can be one of the following:</p> <p><b>base</b> searches only the entry specified in the <b>-b</b> option or defined by the <b>LDAP_BASEDN</b> environment variable.</p> <p><b>one</b> searches only the immediate children of the entry specified in the <b>-b</b> option. Only the children are searched; the actual entry specified in the <b>-b</b> option is not searched.</p> <p><b>sub</b> searches the entry specified in the <b>-b</b> option and all of its descendants; that is, perform a subtree search starting at the point identified in the <b>-b</b> option. This is the default.</p>

Option	Description
-W	Prompt for the password. If this option is not set, anonymous access is used.  Alternatively, use the <b>-w</b> option to pass the password to the utility. Note that the password can be visible in the process list for other users and is saved in the shell's history.
-x	Disables the default SASL connection to allow simple binds.
-Y <i>SASL_mechanism</i>	Sets the SASL mechanism to use for authentication. If no mechanism is set, <b>Idapsearch</b> selects the best mechanism supported by the server.  If <b>-x</b> is not used, then the <b>-Y</b> option must be used.
-z <i>number</i>	Sets the maximum number of entries to return in a response to a search request. This value overwrites the server-side <b>nsslapd-sizelimit</b> parameter when binding using the root DN. <i>wibrown&gt;</i>

#### 14.1.3. Using Special Characters

When using the **Idapsearch** command-line utility, it may be necessary to specify values that contain characters that have special meaning to the command-line interpreter, such as space (' '), asterisk (\*), or backslash (\). Enclose the value which has the special character in quotation marks (""). For example:

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

Depending on the command-line interpreter, use either single or double quotation marks. In general, use single quotation marks ('') to enclose values. Use double quotation marks ("") to allow variable interpolation if there are shell variables. Refer to the operating system documentation for more information.

## 14.2. LDAP SEARCH FILTERS

Search filters select the entries to be returned for a search operation. They are most commonly used with the **Idapsearch** command-line utility. When using **Idapsearch**, there can be multiple search filters in a file, with each filter on a separate line in the file, or a search filter can be specified directly on the command line.

The basic syntax of a search filter is:

```
attribute operator value
```

For example:

```
buildingname>=alpha
```

In this example, **buildingname** is the attribute, **>=** is the operator, and **alpha** is the value. Filters can also be defined that use different attributes combined together with Boolean operators.



## NOTE

When performing a substring search using a matching rule filter, use the asterisk (\*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter **I** and ends with the letter **n**, enter a **I\*n** in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter **u**, enter a value of **u\*** in the value portion of the search filter.

To search for a value that contains the asterisk (\*) character, the asterisk must be escaped with the designated escape sequence, \5c2a. For example, to search for all employees with **businessCategory** attribute values of **Example\*Net product line**, enter the following value in the search filter:

**Example\5c2a\*Net product line**



## NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=user\_name,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless the **dc** attribute exists in this entry and is set to **example**.

### 14.2.1. Using Attributes in Search Filters

The most basic sort of search looks for the presence of attributes or specific values in entries. There are many variations on *how* to look for attributes in entries. It is possible to check that the attribute merely exists, to match an exact value, or to list matches against a partial value.

A presence search uses a wild card (an asterisk) to return every entry which has that attribute set, regardless of value. For example, this returns every entry which has a **manager** attribute:

**"(manager=\*)"**

It is also possible to search for an attribute with a specific value; this is called an *equality* search. For example:

**"(cn=example)"**

This search filter returns all entries that contain the common name set to **example**. Most of the time, equality searches are not case sensitive.

When an attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match the "**(cn=example)**" filter:

**cn: example**  
**cn;lang-fr: example**

It is also possible to search for a partial match on an attribute value, a *substring* index. For example:

```
"(description=*X.500*)"
"(sn=*nderson)"
"(givenname=car*)"
```

The length of the substring searches is configured in the substring index itself, as described in [Section 13.6, “Changing the Width for Indexed Substring Searches”](#).

### 14.2.2. Using Operators in Search Filters

Operators in search filters set the relationship between the attribute and the given search value. For people searches, operators can be used to set a range, to return a last names within a subset of letters in the alphabet or employee numbers that come after a certain number.

```
"(employeeNumber>=500)"
"(sn~=suret)"
"(salary<=150000)"
```

Operators also enable phonetic and approximate searches, which allow more effective searches with imperfect information and are particularly useful in internationalized directories.

The operators that can be used in search filters are listed in [Table 14.1, “Search Filter Operators”](#). In addition to these search filters, special filters can be specified to work with a preferred language collation order. For information on how to search a directory with international character sets, see [Section D.4, “Searching an Internationalized Directory”](#).

**Table 14.1. Search Filter Operators**

Search Type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, <b>cn=example</b>
Substring	= <i>string*</i> <i>string</i>	Returns entries containing attributes containing the specified substring. For example, <b>cn=exa*I</b> . 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, <b>uidNumber &gt;= 5000</b> .
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, <b>uidNumber &lt;= 5000</b> .
Presence	=*	Returns entries containing one or more values for the specified attribute. For example, <b>cn=*</b> .
Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example, <b>I~=san francisco</b> can return <b>I=san francisco</b> .

### 14.2.3. Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

**(Boolean-operator(filter)(filter)(filter)…)**

*Boolean-operator* is any one of the Boolean operators listed in [Table 14.2, “Search Filter Boolean Operators”](#).

For example, this filter returns all entries that do not contain the specified value:

**(!(objectClass=person))**

Obviously, compound search filters are most useful when they are nested together into completed expressions:

**(Boolean-operator(filter)((Boolean-operator(filter)(filter)))**

These compound filters can be combined with other types of searches (approximate, substring, other operators) to get very detailed results. For example, this filter returns all entries whose organizational unit is **Marketing** and whose **description** attribute does not contain the substring **X.500**:

**(&(ou=Marketing)(!(description=\*X.500\*)))**

That filter can be expanded to return entries whose organizational unit is **Marketing**, that do not have the substring **X.500**, and that have **example** or **demo** set as a **manager**:

**(&(ou=Marketing)(!(description=\*X.500\*))( |  
(manager=cn=example,ou=Marketing,dc=example,dc=com)  
(manager=cn=demo,ou=Marketing,dc=example,dc=com)))**

This filter returns all entries that do not represent a person and whose common name is similar to **printer3b**:

**(&(!!(objectClass=person))(cn~=printer3b))**

**Table 14.2. Search Filter Boolean Operators**

Operator	Symbol	Description
AND	&	All specified filters must be true for the statement to be true. For example, <b>(&amp;(filter)(filter)(filter)…)</b> .
OR		At least one specified filter must be true for the statement to be true. For example, <b>(( filter)(filter)(filter)…)</b>
NOT	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the <b>NOT</b> operator. For example, <b>(!filter)</b> .

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.

#### 14.2.4. Using Matching Rules

A *matching rule* tells the Directory Server how to compare two values (the value stored in the attribute and the value in the search filter). A matching rule also defines how to generate index keys. Matching rules are somewhat related to attribute syntaxes. Syntaxes define *the format* of an attribute value; matching rules define how that format is compared and indexed.

There are three different types of matching rules:

- **EQUALITY** specifies how to compare two values for an equal match. For example, how to handle strings like "Fred" and "FRED". Search filters that test for equality (for example, *attribute=value*) use the EQUALITY rule. Equality (eq) indexes use the EQUALITY rule to generate the index keys. Update operations use the EQUALITY rule to compare values to be updated with values already in an entry.
- **ORDERING** specifies how to compare two values to see if one value is greater or less than another value. Search filters that set a range (for example, *attribute<=value* or *attribute>=value*) use the ORDERING rule. An index for an attribute with an ORDERING rule orders the equality values.
- **SUBSTR** specifies how to do substring matching. Substring search filters (for example, *attribute=\*partial\_string\** or *attribute=\*end\_string*) use the SUBSTR rule. Substring (sub) indexes use the SUBSTR rule to generate the index keys.



#### IMPORTANT

A matching rule is required in order to support searching or indexing for the corresponding search filter or index type. For example, an attribute must have an EQUALITY matching rule in order to support equality search filters and eq indexes for that attribute. An attribute must have both an ORDERING matching rule and an EQUALITY matching rule in order to support range search filters and indexed range searches.

A search operation will be rejected with PROTOCOL\_ERROR or UNWILLING\_TO\_PERFORM if an attempt is made to use a search filter for an attribute that has no corresponding matching rule.

#### Example 14.1. Matching Rules and Custom Attributes

Example Corp. administrators create a custom attribute type called ***MyFirstName*** with IA5 String (7-bit ASCII) syntax and an EQUALITY matching rule of caseExactIA5Match. An entry with a ***MyFirstName*** value of **Fred** is returned in a search with a filter of **(*MyFirstName*=Fred)**, but it is not returned for filters like **(*MyFirstName*=FRED)** and **(*MyFirstName*=fred)**. Fred, FRED, and fred are all valid IA5 String values, but they do not match using the caseExactIA5Match rule.

For all three variants of Fred to be returned in a search, then the ***MyFirstName*** should be defined to use the caseIgnoreIA5Match matching rule.

An extensible matching rule search filter can be used to search for an attribute value with a different matching rule than the one defined for the attribute. The matching rule must be compatible with the syntax of the attribute being searched. For example, to run a case insensitive search for an attribute that has a case-sensitive matching rule defined for it, specify a case insensitive matching rule in the search filter.

(MyFirstName:caseIgnoreIA5Match:=fred)



### NOTE

Matching rules are used for searches in internationalized directories, to specify the language types to use for the results. This is covered in [Section D.4, "Searching an Internationalized Directory"](#).



### NOTE

An index for an attributes uses whatever matching rules are defined for that attribute in its schema definition. Additional matching rules to use for an index can be configured using the ***nsMatchingRule*** attribute, as in [Section 13.2.1, "Creating Indexes Using the Command Line"](#).

The syntax of the matching rule filter inserts a matching rule name or OID into the search filter:

*attr:matchingRule:=value*

- *attr* is an attribute belonging to entries being searched, such as ***cn*** or ***mail***.
- *matchingRule* is a string that contains the name or OID of the rule to use to match attribute values according to the required syntax.
- *value* is either the attribute value to search for or a relational operator plus the attribute value to search for. The syntax of the value of the filter depends on the matching rule format used.

A matching rule is actually a schema element, and, as with other schema elements is uniquely identified by an object identifier (OID).

Many of the matching rules defined for Red Hat Directory Server relate to language codes and set internationalized collation orders supported by the Directory Server. For example, the OID **2.16.840.1.113730.3.3.2.17.1** identifies the Finnish collation order.



### NOTE

Unlike other schema elements, additional matching rules cannot be added to the Directory Server configuration.

Most of the matching rules list in following list are used for equality indexes. Matching rules with *ordering* in their name are used for ordering indexes, and those with *substring* in their name are used for substring (SUBSTR) indexes. (The matching rules used for international matching and collation orders use a different naming scheme.)

#### Bitwise AND match

Performs bitwise **AND** matches.

OID: 1.2.840.113556.1.4.803

Compatible syntaxes: Typically used with **Integer** and numeric strings. Directory Server converts numeric strings automatically to integer.

### **Bitwise OR match**

Performs bitwise **OR** matches.

OID: 1.2.840.113556.1.4.804

Compatible syntaxes: Typically used with **Integer** and numeric strings. Directory Server converts numeric strings automatically to integer.

### **booleanMatch**

Evaluates whether the values to match are **TRUE** or **FALSE**

OID: 2.5.13.13

Compatible syntaxes: Boolean

### **caseExactIA5Match**

Makes a case-sensitive comparison of values.

OID: 1.3.6.1.4.1.1466.109.114.1

Compatible syntaxes: **IA5** Syntax, URI

### **caseExactMatch**

Makes a case-sensitive comparison of values.

OID: 2.5.13.5

Compatible syntaxes: Directory String, Printable String, OID

### **caseExactOrderingMatch**

Allows case-sensitive ranged searches (less than and greater than).

OID: 2.5.13.6

Compatible syntaxes: Directory String, Printable String, OID

### **caseExactSubstringsMatch**

Performs case-sensitive substring and index searches.

OID: 2.5.13.7

Compatible syntaxes: Directory String, Printable String, OID

### **caseIgnoreIA5Match**

Performs case-insensitive comparisons of values.

OID: 1.3.6.1.4.1.1466.109.114.2

Compatible syntaxes: **IA5 Syntax, URI**

#### **caselgnoreIA5SubstringsMatch**

Performs case-insensitive searches on substrings and indexes.

OID: 1.3.6.1.4.1.1466.109.114.3

Compatible syntaxes: **IA5 Syntax, URI**

#### **caselgnoreListMatch**

Performs case-insensitive comparisons of values.

OID: 2.5.13.11

Compatible syntaxes: Postal address

#### **caselgnoreListSubstringsMatch**

Performs case-insensitive searches on substrings and indexes.

OID: 2.5.13.12

Compatible syntaxes: Postal address

#### **caselgnoreMatch**

Performs case-insensitive comparisons of values.

OID: 2.5.13.2

Compatible syntaxes: Directory String, Printable String, OID

#### **caselgnoreOrderingMatch**

Allows case-insensitive ranged searches (less than and greater than).

OID: 2.5.13.3

Compatible syntaxes: Directory String, Printable String, OID

#### **caselgnoreSubstringsMatch**

Performs case-insensitive searches on substrings and indexes.

OID: 2.5.13.4

Compatible syntaxes: Directory String, Printable String, OID

#### **distinguishedNameMatch**

Compares distinguished name values.

OID: 2.5.13.1

Compatible syntaxes: Distinguished name (DN)

#### **generalizedTimeMatch**

Compares values that are in a Generalized Time format.

OID: 2.5.13.27

Compatible syntaxes: Generalized Time

#### **generalizedTimeOrderingMatch**

Allows ranged searches (less than and greater than) on values that are in a Generalized Time format.

OID: 2.5.13.28

Compatible syntaxes: Generalized Time

#### **integerMatch**

Evaluates integer values.

OID: 2.5.13.14

Compatible syntaxes: Integer

#### **integerOrderingMatch**

Allows ranged searches (less than and greater than) on integer values.

OID: 2.5.13.15

Compatible syntaxes: Integer

#### **keywordMatch**

Compares the given search value to a string in an attribute value.

OID: 2.5.13.33

Compatible syntaxes: Directory String

#### **numericStringMatch**

Compares more general numeric values.

OID: 2.5.13.8

Compatible syntaxes: Numeric String

#### **numericStringOrderingMatch**

Allows ranged searches (less than and greater than) on more general numeric values.

OID: 2.5.13.9

Compatible syntaxes: Numeric String

#### **numericStringSubstringMatch**

Compares more general numeric values.

OID: 2.5.13.10

Compatible syntaxes: Numeric String

#### **objectIdentifierMatch**

C.compares object identifier (OID) values.

OID: 2.5.13.0

Compatible syntaxes: OID

#### **octetStringMatch**

Evaluates octet string values.

OID: 2.5.13.17

Compatible syntaxes: Octet String

#### **octetStringOrderingMatch**

Supports ranged searches (less than and greater than) on a series of octet string values.

OID: 2.5.13.18

Compatible syntaxes: Octet String

#### **telephoneNumberMatch**

Evaluates telephone number values.

OID: 2.5.13.20

Compatible syntaxes: Telephone Number

#### **telephoneNumberSubstringsMatch**

Performs substring and index searches on telephone number values.

OID: 2.5.13.21

Compatible syntaxes: Telephone Number

#### **uniqueMemberMatch**

Compares both name and UID values.

OID: 2.5.13.23

Compatible syntaxes: Name and Optional UID

#### **wordMatch**

Compares the given search value to a string in an attribute value. This matching rule is case-insensitive.

OID: 2.5.13.32

Compatible syntaxes: Directory String

**Table 14.3. Language Ordering Matching Rules**

<b>Matching Rule</b>	<b>Object Identifiers (OIDs)</b>
English (Case Exact Ordering Match)	2.16.840.1.113730.3.3.2.11.3
Albanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.44.1
Arabic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.1.1
Belorussian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.2.1
Bulgarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.3.1
Catalan (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.4.1
Chinese - Simplified (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.49.1
Chinese - Traditional (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.50.1
Croatian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.22.1
Czech (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.5.1
Danish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.6.1
Dutch (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.33.1
Dutch - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.34.1
English - US (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.11.1
English - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.12.1
English - Irish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.14.1
Estonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.16.1
Finnish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.17.1
French (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.18.1
French - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.19.1
French - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.20.1

Matching Rule	Object Identifiers (OIDs)
French - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.21.1
German (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.7.1
German - Austrian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.8.1
German - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.9.1
Greek (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.10.1
Hebrew (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.27.1
Hungarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.23.1
Icelandic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.24.1
Italian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.25.1
Italian - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.26.1
Japanese (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.28.1
Korean (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.31.1
Lithuanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.30.1
Macedonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.32.1
Norwegian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.35.1
Norwegian - Bokmul (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.36.1
Norwegian - Nynorsk (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.37.1
Polish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.38.1
Romanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.39.1
Russian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.40.1
Serbian - Cyrillic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.45.1

Matching Rule	Object Identifiers (OIDs)
Serbian - Latin (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.41.1
Slovak (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.42.1
Slovenian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.43.1
Spanish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.15.1
Swedish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.46.1
Turkish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.47.1
Ukrainian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.48.1

**Table 14.4. Language Substring Matching Rules**

Matching Rule	Object Identifiers (OIDs)
English (Case Exact Substring Match)	2.16.840.1.113730.3.3.2.11.3.6
Albanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.44.1.6
Arabic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.1.1.6
Belorussian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.2.1.6
Bulgarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.3.1.6
Catalan (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.4.1.6
Chinese - Simplified (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.49.1.6
Chinese - Traditional (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.50.1.6
Croatian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.22.1.6
Czech (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.5.1.6
Danish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.6.1.6
Dutch (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.33.1.6
Dutch - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.34.1.6

Matching Rule	Object Identifiers (OIDs)
English - US (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.11.1.6
English - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.12.1.6
English - Irish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.14.1.6
Estonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.16.1.6
Finnish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.17.1.6
French (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.18.1.6
French - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.19.1.6
French - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.20.1.6
French - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.21.1.6
German (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.7.1.6
German - Austrian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.8.1.6
German - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.9.1.6
Greek (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.10.1.6
Hebrew (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.27.1.6
Hungarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.23.1.6
Icelandic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.24.1.6
Italian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.25.1.6
Italian - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.26.1.6
Japanese (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.28.1.6
Korean (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.29.1.6
Latvian, Lettish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.31.1.6

Matching Rule	Object Identifiers (OIDs)
Lithuanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.30.1.6
Macedonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.32.1.6
Norwegian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.35.1.6
Norwegian - Bokmul (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.36.1.6
Norwegian - Nynorsk (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.37.1.6
Polish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.38.1.6
Romanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.39.1.6
Russian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.40.1.6
Serbian - Cyrillic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.45.1.6
Serbian - Latin (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.41.1.6
Slovak (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.42.1.6
Slovenian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.43.1.6
Spanish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.15.1.6
Swedish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.46.1.6
Turkish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.47.1.6
Ukrainian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.48.1.6

## 14.3. EXAMPLES OF COMMON LDAPSEARCHES

The next set of examples assumes the following:

- The search is for all entries in the directory.
- The directory is configured to support anonymous access for search and read. This means that no bind information has to be supplied in order to perform the search. For more information on anonymous access, see [Section 18.10.1.1.3, “Granting Anonymous Access”](#).
- The server is located on a host named **server.example.com**.

- The server uses port number **389**. Since this is the default port, the port number does not have to be sent in the search request.
- TLS is enabled for the server on port **636** (the default LDAPS port number).
- The suffix under which all data are stored is **dc=example,dc=com**.

### 14.3.1. Returning All Entries

Given the previous information, the following call will return all entries in the directory (subject to the configured size and time resource limits):

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)"
```

**"objectclass=\*"** is a search filter that matches any entry in the directory. Since every entry must have an object class, and the **objectclass** attribute is always indexed, this is a useful search filter to return every entry.

### 14.3.2. Specifying Search Filters on the Command Line

A search filter can be specified directly on the command line as long as the filter is enclosed in quotation marks ("filter"). If the filter is supplied with the command, do not specify the **-f** option. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "cn=babs jensen"
```

### 14.3.3. Searching the Root DSE Entry

The root DSE is a special entry that contains information about the directory server instance, including all of the suffixes supported by the local Directory Server. This entry can be searched by supplying a search base of "", a search scope of **base**, and a filter of **"objectclass=\*"**. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -b "" -s base
"objectclass=*"
```

### 14.3.4. Searching the Schema Entry

The **cn=schema** entry is a special entry that contains information about the directory schema, such as object classes and attribute types.

The following command lists the content of the **cn=schema** entry:

```
# ldapsearch -o ldif-wrap=no -D "cn=Directory Manager" -W -b "cn=schema" \
'(objectClass=subSchema)' -s sub objectClasses attributeTypes matchingRules \
matchingRuleUse dITStructureRules nameForms ITContentRules ldapSyntaxes
```

### 14.3.5. Using LDAP\_BASEDN

To make searching easier, it is possible to set the search base using the **LDAP\_BASEDN** environment variable. Doing this means that the search base does not have to be set with the **-b** option. For information on how to set environment variables, see the documentation for the operating system.

Typically, set **LDAP\_BASEDN** to the directory's suffix value. Since the directory suffix is equal to the root, or topmost, entry in the directory, this causes all searches to begin from the directory's root entry.

For example, set **LDAP\_BASEDN** to **dc=example,dc=com** and search for **cn=babs jensen** in the directory, use the following command-line call:

```
# export LDAP_BASEDN="dc=example,dc=com"
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x "cn=babs jensen"
```

In this example, the default scope of **sub** is used because the **-s** option was not used to specify the scope.

#### 14.3.6. Displaying Subsets of Attributes

The **ldapsearch** command returns all search results in LDIF format. By default, **ldapsearch** returns the entry's distinguished name and all of the attributes that a user is allowed to read. The directory access control can be set such that users are allowed to read only a subset of the attributes on any given directory entry. Only operational attributes are not returned. For operational attributes to be returned as a result of a search operation, explicitly specify them in the search command or use **+** to return all operational attributes.

It may not be necessary to have all of the attributes for an entry returned in the search results. The returned attributes can be limited to just a few specific attributes by specifying the required ones on the command line immediately after the search filter. For example, to show the **cn** and **sn** attributes for every entry in the directory, use the following command-line call:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" sn cn
```

#### 14.3.7. Searching for Operational Attributes

Operational attributes are special attributes set by the Directory Server itself that are used by the server to perform maintenance tasks, like processing access control instructions. They also show specific information about the entry, like the time it was initially created and the name of the user who created it. Operational attributes are available for use on every entry in the directory, regardless of whether the attribute is specifically defined for the object class of the entry.

Operational attributes are not returned in regular **ldapsearches**. According to [RFC3673](#), use **+** to return all operational attributes in a search request:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" '+'
```

To return only some defined operational attributes, explicitly specify them in the **ldapsearch** request:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" creatorsName createTimeStamp modifiersName modifyTimeStamp
```

The complete list of operational attributes is in the "Operational Attributes and Object Classes" chapter in the [Red Hat Directory Server 11 Configuration, Command, and File Reference](#).



## NOTE

To return all of the regular entry attributes along with the specified operational attributes, use the special search attribute, “\*\*”, in addition to the operational attributes that are listed.

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b  
"dc=example,dc=com" -s sub -x "(objectclass=*)" "*" aci
```

The asterisk must be enclosed in quotation marks to prevent it from being interpreted by the shell.

### 14.3.8. Specifying Search Filters Using a File

Search filters can be entered into a file instead of entering them on the command line. In this case, specify each search filter on a separate line in the file. The **ldapsearch** command runs each search in the order in which it appears in the file.

For example:

```
sn=example  
givenname=user
```

**ldapsearch** first finds all the entries with the surname set to **example**, then all the entries with the **givenname** set to **user**. If an entry is found that matches both search criteria, then the entry is returned twice.

For example, in this search, the filters are specified in a file named **searchdb**:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -f searchdb
```

The set of attributes returned here can be limited by specifying the attribute names at the end of the search line. For example, the following **ldapsearch** command performs both searches but returns only the DN and the **givenname** and **sn** attributes of each entry:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -f searchdb sn  
givenname
```

### 14.3.9. Specifying DNs That Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, the comma must be escaped with a backslash (\). For example, to find everyone in the **example.com Bolivia, S.A.** subtree, use the following command:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -s base -b  
"l=Bolivia\,S.A.,dc=example,dc=com" "objectclass=*"
```

### 14.3.10. Using a Client Certificate to Bind to Directory Server

See [Section 9.9.4, “Authenticating Using a Certificate”](#).

### 14.3.11. Searching with Language Matching Rules

To explicitly submit a matching rule in a search filter, insert the matching rule after the attribute:

`attr:matchingRule:=value`

Matching rules are frequently used for searching internationalized directories. For example, this searches for the department numbers after N4709 in the Swedish (**2.16.840.1.113730.3.3.2.46.1**) matching rule.

`departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709`

More examples of performing internationalized searches are given in [Section D.4, “Searching an Internationalized Directory”](#).

#### 14.3.12. Searching for Attributes with Bit Field Values

Bitwise searches use the bitwise AND or bitwise OR matching rules to perform bitwise search operations on attributes with values that are bit fields.



##### NOTE

Attributes with values for bit fields are not common in LDAP. (No default Directory Server schema use bit fields as attribute syntax.) However, several LDAP syntaxes support integer-style values. Custom attributes can be defined which use bit field values, and applications can use those custom attributes to perform bitwise operations against bit field values.

The bitwise AND matching rule (**1.2.840.113556.1.4.803**) checks that the bit given in the assertion value is set in the bit field attribute value. (This is somewhat analogous to an equality search.) In this example, the userAccountControl value must be set to the bit representing 2.

`"(UserAccountControl:1.2.840.113556.1.4.803:=2)"`

In this example, the userAccountControl value must have all of the bits set that are set in the value 6 (bits 2 and 4).

`"(UserAccountControl:1.2.840.113556.1.4.803:=6)"`

The bitwise OR matching rule (**1.2.840.113556.1.4.804**) checks to see if *any* of the bits in the assertion string are represented in the attribute value. (This is somewhat analogous to a substring search.) In this example, the userAccountControl value must have any of the bits which are set in the bit field of 6, meaning that the attribute value can be 2, 4, or 6.

`"(UserAccountControl:1.2.840.113556.1.4.804:=6)"`

Bitwise searches can be used with Windows-Red Hat Enterprise Linux integration, such as using Samba file servers.

### 14.4. IMPROVING SEARCH PERFORMANCE THROUGH RESOURCE LIMITS

With large directories, searching through every entry in the database can have a negative impact on the server performance. Effective indexing can improve the performance in certain scenarios. However, in large databases, this may still not reduce the search scope enough to improve the performance.

Reasonable limits can be set on user and client accounts to reduce the total number of entries or the total amount of time spent in an individual search, which both makes searches more responsive and improves overall server performance.

Server limits for search operations are controlled using special operational attribute values on the client application binding to the directory. You can set the following search operation limits:

- *Look through limit*. Specifies how many entries can be examined for a search operation.
- *Size limit*. Specifies the maximum number of entries the server returns to a client application in response to a search operation.
- *Time limit*. Specifies the maximum time the server spends processing a search operation.
- *Idle timeout*. Specifies the time a connection to the server can be idle before the connection is dropped.
- *Range timeout*. Specifies a separate look-through limit specifically for searches using a range.

The resource limits set for the client application take precedence over the default resource limits set for in the global server configuration.



#### NOTE

The Directory Manager receives unlimited resources by default, with the exception of range searches.

#### 14.4.1. Search Performance and Resource Limits

For details, see the corresponding section in the [Red Hat Directory Server Performance Tuning Guide](#).

#### 14.4.2. Fine Grained ID List Size

For details, see the corresponding section in the [Red Hat Directory Server Performance Tuning Guide](#).

#### 14.4.3. Setting User and Global Resource Limits Using the Command Line

Through the command line, administrators can set user-level resource limits, global resource limits, and limits for specific kinds of searches, such as simple paged and range searches. [Section 13.1.3, “Overview of the Searching Algorithm”](#) has more information on how these resource limits affect Directory Server search performance.

[Section 14.4.3, “Setting User and Global Resource Limits Using the Command Line”](#) lists operational attributes which can be set for each entry using the command line. Use **ldapmodify** to add the attributes to the entry.

User-level attributes are set on the individual entries, while global configuration attributes are set in the appropriate server configuration area.

##### Look-through limit

Specifies how many entries are examined for a search operation. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: **nsLookThroughLimit**

- Global configuration:
  - Attribute: ***nsslapd-lookthroughlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

### Page look-through limit

As with the look-through limit, specifies how many entries are examined, but specifically for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsPagedLookThroughLimit***
- Global configuration:
  - Attribute: ***nsslapd-pagedlookthroughlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

### Size limit

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.

- User-level attribute: ***nsSizeLimit***
- Global configuration:
  - Attribute: ***nsslapd-sizelimit***
  - Entry: **cn=config**

### Paged size limit

As with the size limit, specifies the maximum number of entries the server returns to a client application but only for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsPagedSizeLimit***
- Global configuration:
  - Attribute: ***nsslapd-pagedsizelimit***
  - Entry: **cn=config**

### Time Limit

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.

- User-level attribute: ***nsTimeLimit***
- Global configuration:
  - Attribute: ***nsslapd-timelimit***
  - Entry: **cn=config**

## Idle timeout

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.

- User-level attribute: ***nsidletimeout***
- Global configuration:
  - Attribute: ***nsslapd-idletimeout***
  - Entry: **cn=config**

## ID list scan limit

Specifies the maximum number of entry IDs loaded from an index file for search results. If the ID list size is greater than this value, the search will not use the index list but will treat the search as an unindexed search and look through the entire database.

- User-level attribute: ***nsIDListScanLimit***
- Global configuration:
  - Attribute: ***nsslapd-idlistscanlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

## Paged ID list scan limit

As with the ID list scan limit, specifies the maximum number of entry IDs loaded from an index file for search results, but specifically for paged search operations.

- User-level attribute: ***nsPagedIDListScanLimit***
- Global configuration:
  - Attribute: ***nsslapd-pagedidlistscanlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

## Range look-through limit

Specifies how many entries are examined for a range search operation (a search using greater-than, equal-to-or-greater-than, less-than, or equal-to-less-than operators). Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: not available
- Global configuration:
  - Attribute: ***nsslapd-rangefollowthroughlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

For information about the parameters listed above, see their descriptions in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

For example, this sets the size limit for a user:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: uid=user_name,ou=People,dc=example,dc=com
changetype: modify
add: nsSizeLimit
nsSizeLimit: 500
```

The **ldapmodify** statement adds the ***nsSizeLimit*** attribute to the user's entry and gives it a search return size limit of 500 entries.



#### NOTE

Set an access control list (ACL) to prevent users changing the setting. For details about ACLs, see [Chapter 18, Managing Access Control](#).

#### 14.4.4. Setting Resource Limits on Anonymous Binds

Resource limits are set on a user entry. An anonymous bind, obviously, does not have a user entry associated with it. This means that the global resource limits usually apply to anonymous operations. However, it is possible to configure resource limits specifically for anonymous binds by creating a template user entry that has resource limits, and then applying that template to anonymous binds.

1. Create a template entry and set whatever resource limits you want to apply to anonymous binds.



#### NOTE

For performance reasons, the template should be in the normal back end, not in the ***cn=config*** suffix, which does not use an entry cache.

For example:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=anonymous_template,ou=people,dc=example,dc=com
objectclass: nsContainer
objectclass: top
cn: anonymous_template
nsSizeLimit: 250
nsLookThroughLimit: 1000
nsTimeLimit: 60
```

2. On all suppliers in a replication topology, add the ***nsslapd-anonlimitsdn*** parameter to the server configuration, pointing to the DN of the template entry. Any of the resource limits in [Section 14.4.3, "Setting User and Global Resource Limits Using the Command Line"](#) can be set.  
For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
anonlimitsdn="cn=anonymous_template,ou=people,dc=example,dc=com"
```

#### 14.4.5. Improving Performance for Range Searches

Range searches use operators (Section 14.2.2, “Using Operators in Search Filters” ) to set a bracket to search for and return an entire subset of entries within the directory. For example, this searches for every entry modified at or after midnight on January 1:

```
(modifyTimestamp>=20210101010101Z)
```

The nature of a range search is that it must evaluate every single entry within the directory to see if it is within the range given. Essentially, a range search is always an all IDs search.

For most users, the look-through limit kicks in and prevents range searches from turning into an all IDs search. This improves overall performance and speeds up range search results. However, some clients or administrative users like Directory Manager may not have a look-through limit set. In that case, a range search can take several minutes to complete or even continue indefinitely.

It is possible to set a separate range look-through limit. This allows clients and administrative users to have high look-through limits while still allowing a reasonable limit to be set on potentially performance-impaired range searches.

This is configured in the **nsslapd-rangelookthroughlimit** attribute. The default value is **5000**, the same as the default **nsslapd-lookthroughlimit** attribute value.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: add
add: nsslapd-rangelookthroughlimit
nsslapd-rangelookthroughlimit: 7500
```

## 14.5. USING PERSISTENT SEARCH

A persistent search is an **ldapsearch** which remains open even after the initial search results are returned.



### IMPORTANT

The OpenLDAP client tools with Red Hat Enterprise Linux do not support persistent searches. The server itself, however, does. Other LDAP clients must be used to perform persistent searches.

The purpose of a persistent search is to provide a continuous list of changes to the directory entries as well as the complete entries themselves, something like a hybrid search and changelog. Therefore, the search command must specify what entries to return (the search parameters) and what changes cause an entry to be returned (entry change parameters).

Persistent searches are especially useful for applications or clients which access the Directory Server and provide two important benefits:

- Keep a consistent and current local cache.

Any client will query local cache before trying to connect to and query the directory. Persistent searches provide the local cache necessary to improve performance for these clients.

- Automatically initiate directory actions.

The persistent cache can be automatically updated as entries are modified, and the persistent search results can display what kind of modification was performed on the entry. Another application can use that output to update entries automatically, such as automatically creating an email account on a mail server for new users or generating a unique user ID number.

There are some performance considerations when running persistent searches, as well:

- The **ldapsearch** does not send a notification when the client disconnects, and the change notifications are not sent for any changes made while the search is disconnected. This means that the client's cache will not be updated if it is ever disconnected and there is no good way to update the cache with any new, modified, or deleted entries that were changed while it was disconnected.
- An attacker could open a large number of persistent searches to launch a denial of service attack.
- A persistent search requires leaving open a TCP connection between the Directory Server and client. This should only be done if the server is configured to allow a lot of client connections and has a way to close idle connections.

In the access logs, a persistent search is identified with the tag **options=persistent**.

```
[12/Jan/2009:12:51:54.899423510 -0500] conn=19636710736396323 op=0 SRCH
base="dc=example,dc=com" scope=2 filter="(objectClass=person)" attrs=ALL options=persistent
```

## 14.6. SEARCHING WITH SPECIFIED CONTROLS

The Directory Server has defined controls in its **supportedControls** attribute in its DSE. Some of these define server operations like replication; other are allowed extended operations like get effective rights or dereferencing controls which clients can pass through LDAP operations to the server.

These controls can be specified using the **-E** option by giving the control OID, its criticality for the **ldapsearch**, and any information required for the control operation.

```
-E '[!]control_OID:control_information'
```

Some controls, like server-side sorting and simple paged results, have an alias that can be used to pass the control to the search operation. When the control alias is used, then the results are formatted, since the control is recognized by the client.

### 14.6.1. Retrieving Effective User Rights

A get effective-rights search control is passed using the control OID. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x -E '!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=jsmith,ou=people,dc=example,dc=com'
(objectclass=*)"
```



#### IMPORTANT

When a control is passed with its OID, the results from the search are unformatted.

Get effective rights searches are covered in much more detail in the access control chapter, [Section 18.11, “Checking Access Rights on Entries \(Get Effective Rights\)”](#).

## 14.6.2. Using Server-Side Sorting

Server-side sorting is performed as other control operations, using the **-E** flag and the **sss** control alias. The structure of the operation sets the attribute by which to sort the results and, optionally, the sort order and ordering rule.

**-E sss=[-]attribute\_name:[ordering\_rule\_OID]**

The dash (-) is an optional flag that reverses the sort order, which naturally runs descending. The matching rule tables in [Section 14.2.4, “Using Matching Rules”](#) contain the ordering rules supported by the Directory Server.

For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x -E sss=-uidNumber:2.5.13.15 "(objectclass=*)"
```

## 14.6.3. Performing Dereferencing Searches

A *dereferencing* search is a quick way to track back over cross-references in an entry and return information about the referenced entry. For example, a group entry contains references to its member's user entries. A regular search first searches for the group, then lists its members, and then requires a separate search for each member. A dereferencing search for the group entry returns information about the members – such as their locations, email addresses, or managers – *along with* the information for the group, all in a single search request.

Dereferencing simplifies many client operations and reduces the number of search operations that are performed. Cross-links show relationships between entries. Some operations may require getting a list of cross-links from one entry and then performing a series of subsequent searches to get information from each entry on the list. Dereferencing allows those sequences of searches to be consolidated into a single search.



### IMPORTANT

Dereferencing operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support dereferencing searches.

The format of the dereference arguments is:

**-E 'deref=deref\_attribute:list\_of\_attributes'**

The *deref\_attribute* is the attribute in the search target that contains the reference. This can be any attribute which has a DN for its value, such as **member** or **manager**.



### NOTE

Not only must the value of the *deref\_attribute* be a DN, but the actual defined syntax for the attribute must be DN syntax (**1.3.6.1.4.1.1466.115.121.1.12**).

The *list\_of\_attributes* is one or more attributes in the referenced entry which will be returned along with the primary search results. Multiple attributes can be separated by commas, like **I,mail,cn**.

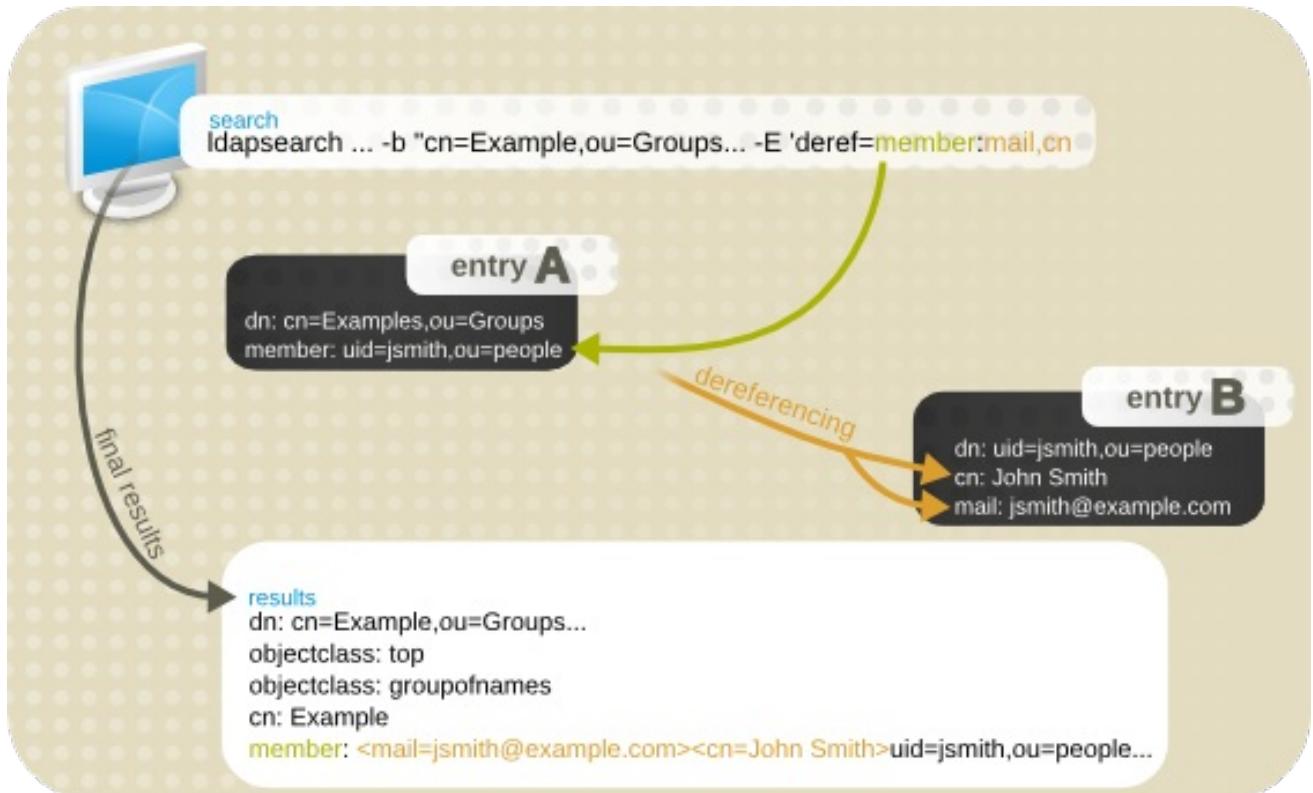


Figure 14.1. Simple Dereferencing Search Command

The requested dereferenced information requested in the search argument is returned with the rest of the search results. For example, this dereferencing search tells the server to use the **member** attribute in the search target entry (the Engineers group) as the *deref\_attribute*. It then returns the locality attribute for each member.

```
# ldapsearch -x -D "cn=Directory Manager" -W -b "cn=Example,ou=Groups,dc=example,dc=com" -E 'deref=member:mail,cn' "(objectclass=*)"

# Engineers, Groups, example.com
dn: cn=Engineers,ou=Groups,dc=example,dc=com
control: 1.3.6.1.4.1.4203.666.5.16 false MIQAAADNMIQAAAA1BAZtZW1iZXIEK2NuPURId

mVsb3BlcnMsIG91PUdyb3VwcywgZGM9ZXhhbXBsZSxkYz1jb20whAAAADIEBm1lbWJlcgQoY249VG

VzdGVycywgb3U9R3JvdXBzLCBkYz1leGFtcGxILGRjPWNvbTCEAAAAVAQGbWVtYmVyBCp1aWQ9Z
W5

nLCBvdT1lbmdpbmVlcmLuZywgZGM9ZXhhbXBsZSxkYz1jb22ghAAAABowhAAAABQEAWwxhAAAAAs
E
CUNhbWJyaWRnZQ==

# member: <mail=jsmith@example.com><cn=John
Smith>;uid=jsmith,ou=people,dc=example,dc=com
objectClass: top
objectClass: inetuser
objectClass: groupofnames
cn: Engineers
member: uid=jsmith,ou=people,dc=example,dc=com
```

#### 14.6.4. Using Simple Paged Results

Search results can be very large, and a part of processing the results is organizing the results. One method of doing this is using *simple paged results*, a control that breaks the results into pages of a certain length.

The simple paged results control sets the number of entries to display at a time. The results can be scrolled through one page at a time which makes the results easier to digest. The full behavior of the control is described in [RFC 2696](#).

Simple paged results are implemented as an LDAP control extension for the Directory Server. Its OID is **1.2.840.113556.1.4.319**.

##### How Simple Paged Results Work

When you start a simple paged results search:

1. The client sends the search to the server, together with the paged results control and with how many records to return in the first page.
2. Before Directory Server starts returning data, the server generates an estimate how many records can be returned in total.

The estimate of records is not an exact number. The total number of records returned can be lower than the estimate. The reasons for such a scenario include

- attributes used in the search filter do not exist in the index. For an optimal result, all queried attributes must be indexed.
- before an entry is send to the client, access control lists (ACL) are validated. Insufficient permissions can prevent the entry from being returned.

After generating the estimate, the server sends the first set of results, a cookie, and the estimated number of records.

3. The returned records are displayed in the client. The user can now enter how many records should be returned in the next request. The requested number is now sent, together with the cookie, to the server.
4. The server retrieves the requested number of records from the database and sends them together with a cookie to the client.
5. The previous two steps are repeated until all records are sent or the search is cancelled.

##### Simple Paged Results and OpenLDAP Tools

The format of the simple paged result search option with **ldapsearch** is:

**-E pg=size**

The *size* value is the page size, or the number of entries to include per page. For example:

```
ldapsearch -x -D "cn=Directory Manager" -W -b "ou=Engineers,ou=People,dc=example,dc=com" -E
pg=3 "(objectclass=*)" cn

dn: uid=jsmith,ou=Engineers,ou=People,dc=example,dc=com
cn: John Smith
```

```
dn: uid=bjensen,ou=Engineers,ou=People,dc=example,dc=com
```

cn: Barbara Jensen

```
dn: uid=hmartin,ou=Engineers,ou=People,dc=example,dc=com
```

cn: Henry Martin

Results are sorted.

next page size (3): 5

The tag at the end shows the configured page size (the number in parentheses) from the search. After the colon, one enters the page size for the next page, so entering **5** as shown would open the next page of results with five entries.



## IMPORTANT

Simple paged results operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support simple paged results, such as Perl Net::LDAP.

### Simple Paged Results and Server-Side Sorting

Simple paged results can be used together with server-side sorting. Server-side sorting is a control which performs the sort process on the server rather than in a client; this is usually done for a search which uses a particular matching rule. (This behavior is defined in [RFC 2891](#).) The OpenLDAP client tools do not support server-side sort with the simple paged results control, but other LDAP utilities such as Perl Net::LDAP do support both.

### Multiple Simple Paged Results Requests on a Single Connection

Some clients may open a single connection to the Directory Server, but send multiple operation requests, including multiple search requests using the simple paged results extension.

Directory Server can manage and interpret multiple simple paged searches. Each search is added as an entry in an array. When the paged search request is first sent, there is a cookie created and associated with the search results. Each page of results is returned with that cookie, and that cookie is used to request the next page of results. On the last page, the cookie is empty, signalling the end of the results. This keeps each set of search results separate.

When there are multiple simple paged results on a single connection, the timeout limits are still observed, but *all* open search requests must reach their configured time limit before *any* paged search is disconnected.

### Simple Paged Results, Contrasted with VLV Indexes

VLV indexes are similar to simple paged results in that they also return a usable browsing list of results. The main difference is in how that list is generated. Simple paged results are calculated per search, while VLV indexes are a permanent list. Overall, VLV indexes are faster for searches, but do require some server-side configuration and overhead for the server to maintain.



## NOTE

Simple paged results and VLV indexes *cannot* be used on the same search. Simple paged results would attempt to manipulate the VLV index, which is already a browsing index. If the control is passed for a search using a VLV index, then the server returns an **UNWILLING\_TO\_PERFORM** error.

For more information on VLV indexes, see [Section 13.4, “Creating Browsing Indexes”](#).

### 14.6.5. Pre- and Post-read Entry Response Controls

Red Hat Directory Server supports pre- and post-read entry response controls according to [RFC 4527](#). If a client requests one or both response controls, an LDAP search entry is returned, that contains the attribute's value before and after the update.

When the pre-read control is used, an LDAP search query is returned containing the specified attribute's value before modification. When the post-read control is used, the query contains the attribute's value after modification. Both controls can be used at the same time. For example, to update the ***description*** attribute and display the value before and after the modification:

```
# ldapmodify -D "cn=Directory Manager" -W -x \
-e \!preread=description -e \!postread=description
dn: uid=user,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: new description

modifying entry "uid=user,ou=People,dc=example,dc=com"
control: 1.3.6.1.1.13.1 false ZCkEJXVpZD1qdXNlcixvdT1QZW9wbGUzZGM9ZXhhbXBsZSxk
Yz1jb20wAA==
# ==> preread
dn: uid=user,ou=People,dc=example,dc=com
description: old description
# <== preread
control: 1.3.6.1.1.13.2 false ZEsEJXVpZD1qdXNlcixvdT1QZW9wbGUzZGM9ZXhhbXBsZSxk
Yz1jb20wljAgBAtkZXNjcmIwdGlvbjERBA9uZXcgZGVzY3JpcHRpb24=
# ==> postread
dn: uid=user,ou=People,dc=example,dc=com
description: new description
# <== postread
```

# CHAPTER 15. MANAGING REPLICATION

*Replication* is the mechanism by which directory data is automatically synchronized from one Red Hat Directory Server instance to another; it is an important mechanism for extending the directory service beyond a single server configuration. This chapter describes the tasks to be performed on the supplier and consumer servers to set up single-supplier replication, multi-supplier replication, and cascading replication.

## 15.1. REPLICATION OVERVIEW

Replication is the mechanism by which directory data is automatically synchronized from one Directory Server to another. Updates of any kind – entry additions, modifications, deletion or renaming of an entry – are automatically mirrored to other Directory Servers using replication.

- [Section 15.1.1, “What Directory Units Are Replicated”](#)
- [Section 15.1.2, “Read-Write and Read-Only Replicas”](#)
- [Section 15.1.3, “Suppliers and Consumers”](#)
- [Section 15.1.4, “Changelog”](#)
- [Section 15.1.5, “Replication Identity”](#)
- [Section 15.1.6, “Replication Agreement”](#)

### 15.1.1. What Directory Units Are Replicated

The smallest unit of the directory which can be replicated is a database. This means that one can replicate an entire database but not a subtree within a database. Therefore, when creating the directory tree, consider any replication plans as part of determining how to distribute information.

Replication also requires that one database correspond to one suffix. This means that a suffix (or namespace) that is distributed over two or more databases using custom distribution logic cannot be replicated. For more information on this topic, see [Section 2.2, “Creating and Maintaining Databases”](#).

### 15.1.2. Read-Write and Read-Only Replicas

A database that participates in replication is called a *replica*. There are two kinds of replicas: read-write or read-only. A *read-write replica* contains supplier copies of directory information and can be updated. A *read-only replica* services read, search, and compare requests, but refers all update operations to read-write replicas. A server can hold any number of read-only or read-write replicas.

### 15.1.3. Suppliers and Consumers

A server that holds a replica that is sent to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is received from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica, with two exceptions:

- In the case of cascading replication, the hub server holds a read-only replica that it supplies to consumers. [Section 15.5, “Cascading Replication”](#) has more information.
- In the case of multi-supplier replication, the *suppliers* are both suppliers and consumers for the same information. For more information, see [Section 15.4, “Multi-Supplier Replication”](#).

Replication is always initiated by the supplier server, never by the consumer (*supplier-initiated replication*). Supplier-initiated replication allows a supplier server to be configured to send data to multiple consumer servers.

#### 15.1.4. Changelog

Every supplier server maintains a *changelog*, a record of all changes that a supplier or hub needs to send to its consumers. A changelog is a special kind of database that keeps 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-supplier replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the changelog.

The changelog uses the same database environment as the main database. Implementing the changelog as part of the main database ensures the database and changelog are always synchronized, reduces the required database cache size, and simplifies backup and restore operations.



#### IMPORTANT

The changelog only writes changelog RUV entries to the database when the server is shut down, and otherwise the RUVs are managed in memory. When you back up the database of a supplier, use the **dsctl db2bak** command or the web console. Both ways, the RUVs are written to the database before the backup starts.

In Directory Server, the changelog is only intended for internal use by the server.

#### 15.1.5. Replication Identity

When replication occurs between two servers, the replication process uses a special entry, called the *replication manager* entry, to identify replication protocol exchanges and to control access to the directory data. The replication manager entry, or any entry used during replication, must meet the following criteria:

- It is created on the consumer server in the **cn=config** entry.
- Create this entry on every server that receives updates from another server, meaning on every hub or dedicated consumer.
- When a replica is configured as a consumer or hub, this entry must be specified as the one authorized to perform replication updates.
- The replication agreement is created on the supplier server, the DN of this entry must be specified in the replication agreement.
- This entry, with its special user profile, bypasses all access control rules defined on the consumer server for the database involved in that replication agreement.

#### 15.1.6. Replication Agreement

Directory Servers use replication agreements to define their replication configuration. A replication agreement describes replication between *one* supplier and *one* consumer only. The agreement is configured on the supplier server and must specify all required replication information:

- The database to be replicated.

- The consumer server to which the data is pushed.
- The days and times during which replication can occur.
- The DN and credentials that the supplier server must use to bind (the replication manager entry or supplier bind DN).
- How the connection is secured (TLS, client authentication).
- Any attributes that will not be replicated (fractional replication).

### 15.1.7. Replicating a Subset of Attributes with Fractional Replication

Fractional replication sets a specific subset of attributes that will not be transmitted from a supplier to the consumer (or another supplier). Administrators can therefore replicate a database without replicating all the information that it contains or all of the information in every entry.

Fractional replication is enabled and configured per replication agreement, not per entry. Excluding attributes from replication is applied equally to all entries within the replication agreement's scope.

For attributes that are defined as optional (**MAY** keyword) in the schema, it is possible to set different attributes to be replicated for an incremental update and a total update. The incremental update list (**nsDS5ReplicatedAttributeList**) must always be set to enable fractional replication; if that is the only attribute set, then it applies to both incremental and total updates. The optional **nsDS5ReplicatedAttributeListTotal** attribute sets an additional fractional replication list for total updates. This is described in [Section 15.11.1, "Setting Different Fractional Replication Attributes for Total and Incremental Updates"](#).



#### NOTE

An update to an excluded attribute still triggers a modify event and generates an empty replication update. The **nsds5ReplicaStripAttrs** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attributes like **modifiersName**.

If a replication event is *not* empty, the stripped attributes *are* replicated. These attributes are removed from updates only if the event would otherwise be empty.

### 15.1.8. Replication over TLS

For security reasons, configure Directory Server instances involved in replication to only replicate data over a TLS connection. Note that if attribute encryption is enabled, a secure connection is always required for replication.

#### Prerequisites

Before you can configure replication over TLS, the following prerequisites are required:

- Configure both the supplier and consumer servers to use TLS. See [Section 9.4.1, "Enabling TLS in Directory Server"](#).
- Configure the consumer server to recognize the supplier server's certificate as the supplier DN. Do this only to use TLS client authentication rather than simple authentication. See [Section 9.9, "Using Certificate-based Client Authentication"](#).



## IMPORTANT

Replication configured over TLS with certificate-based authentication fails if the supplier's certificate is only capable of behaving as a server certificate and not also as a client during an TLS handshake. Replication with certificate-based authentication uses the Directory Server's server certificate for authentication to the remote server.

If you use **certutil** to generate the Certificate Signing Request (CSR), pass the **--nsCertType=sslClient,sslServer** option to the command to set the certificate required type.

## Configuring Replication over TLS

For details about configuring replication:

- When using user name and password authentication:
  - [Section 15.3, "Single-supplier Replication"](#)
  - [Section 15.4, "Multi-Supplier Replication"](#)
  - [Section 15.5, "Cascading Replication"](#)
- When using certificate-based authentication, see [Section 15.6, "Configuring Replication Partners to use Certificate-based Authentication"](#).

## 15.2. CONFIGURING BOOTSTRAP CREDENTIALS

When you use bind distinguished name (DN) groups in a replication agreement, there can be situations where the group is not present or outdated:

- During online initialization where you must authenticate to the replica before the database is initialized
- When you use GSSAPI as authentication method and the Kerberos credentials are changed

If you configured bootstrap credentials in a replication agreement, Directory Server uses these credentials in case that the connection failed because one of the following errors:

- **LDAP\_INVALID\_CREDENTIALS (err=49)**
- **LDAP\_INAPPROPRIATE\_AUTH (err=48)**
- **LDAP\_NO\_SUCH\_OBJECT (err=32)**

If the bind succeeds with the bootstrap credentials, the server establishes the replication connection and a new replication session begins. This allows any updates to the bind DN group members to be updated. By default on the next replication session, Directory Server uses the default credentials in the agreement, that now succeeds.

The bootstrap credentials also fail, Directory Server stops trying to connect.

### Procedure

To set the bootstrap credentials when you create a replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt create ... --bootstrap-bind-dn "bind_DN" --bootstrap-bind-passwd "password" --bootstrap-bind-method bind_method --bootstrap-conn-protocol connection_protocol ...
```

To set the bootstrap credentials in an existing replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt set --suffix="suffix" --bootstrap-bind-dn "bind_DN" --bootstrap-bind-passwd "password" --bootstrap-bind-method bind_method --bootstrap-conn-protocol connection_protocol agreement_name
```

### 15.3. SINGLE-SUPPLIER REPLICATION

In a single-supplier replication scenario, the supplier copy of the directory data is held in a single read-write replica on one server called the *supplier server*. The supplier also maintains the changelog for this replica. On another server, called the *consumer server*, a read-only copy of the directory is stored. In a single-supplier replication environment, you can run multiple consumers.

Use a single-supplier replication topology, for example, if a suffix receives a large number of search requests, but a small number of write requests. To distribute the load, clients can run searches for the suffix on all servers in the topology and send write requests to the supplier.

The following diagram shows a single-supplier replication environment with two consumers:

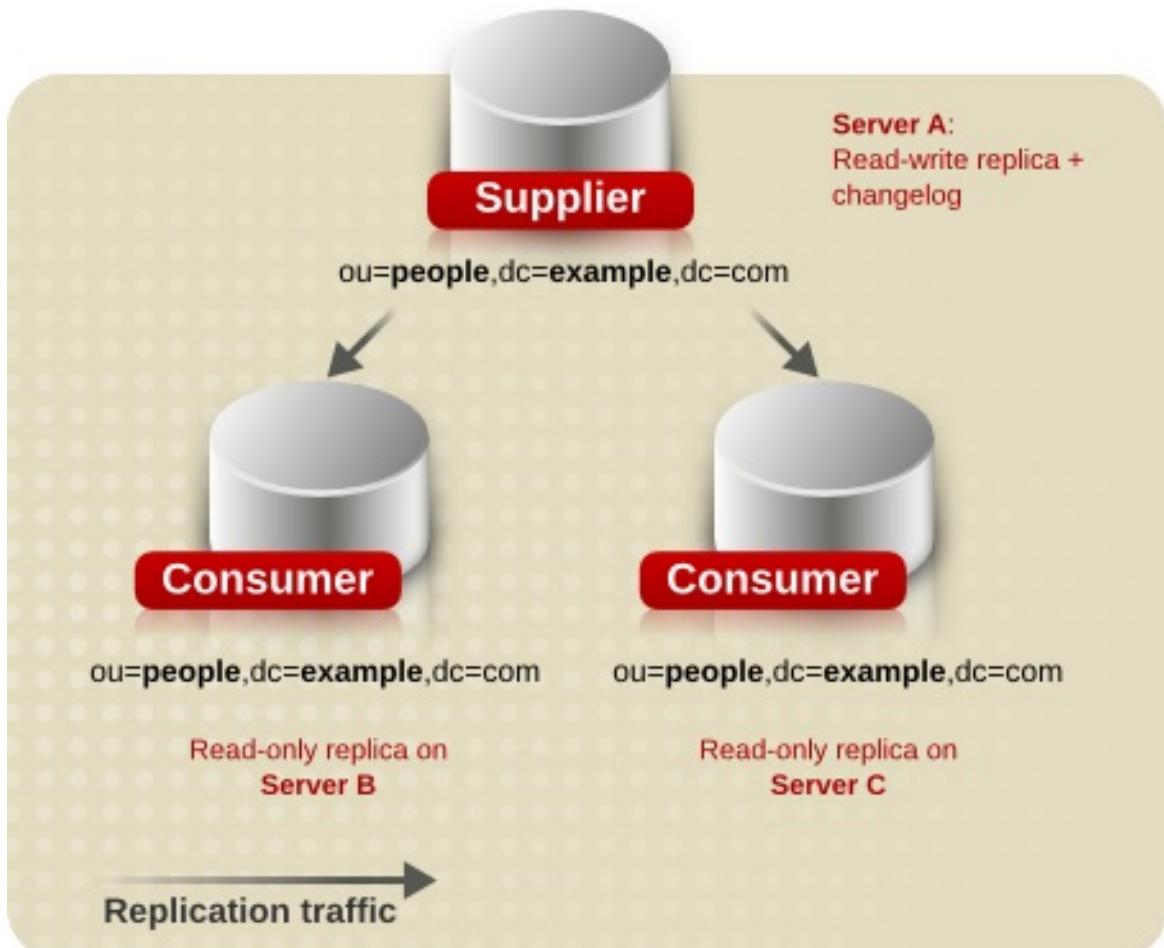


Figure 15.1. Single-supplier Replication

Use the command line or web console to set up a single-supplier replication topology. See:

- Section 15.3.1, "Setting up Single-supplier Replication Using the Command Line"
- Section 15.3.2, "Setting up Single-supplier Replication Using the Web Console"

### 15.3.1. Setting up Single-supplier Replication Using the Command Line

The following example assumes that you have an existing Directory Server instance running on a host named **supplier.example.com** that will act as a supplier in the replication topology to be set up. The following procedures describe how to add a read-only consumer named **consumer.example.com** to the topology, and how to configure single-supplier replication for the **dc=example,dc=com** suffix.

#### Steps to be Performed on the Consumer

On the **consumer.example.com** host:

1. Install Directory Server, and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. In case you created the instance without a database, create the database for the suffix. For example, to create a database named **userRoot** for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://consumer.example.com backend \
create --suffix="dc=example,dc=com" --be-name="userRoot"
```

For details on creating a database for a suffix, see [Section 2.1.1, "Creating Suffixes"](#).

3. Enable replication for the suffix, and create the replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://consumer.example.com replication \
enable --suffix="dc=example,dc=com" --role="consumer" \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```

This command configures the **consumer.example.com** host as a consumer for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password, and allows this account to replicate changes for the suffix to this host.

To add multiple consumers for the suffix to the topology, repeat the steps on each consumer.

#### Steps to be Performed on the Supplier

On the **supplier.example.com** host:

1. Enable replication for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id=1
```

This command configures the **supplier.example.com** host as a supplier for the **dc=example,dc=com** suffix, and sets the replica ID for this entry to **1**.



#### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

2. Add the replication agreement, and initialize the consumer. For example:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="consumer.example.com" --port=636 \
--conn-protocol=LDAPS --bind-dn="cn=replication manager,cn=config" \
--bind-passwd="password" --bind-method=SIMPLE --init \
example-agreement
```

This command creates a replication agreement named **example-agreement**. The replication agreement defines settings, such as the consumer's host name, protocol, and authentication information that the supplier uses when connecting and replicating data to the consumer.

After the agreement was created, Directory Server initializes the consumer. To initialize the consumer later, omit the **--init** option. Note that replication does not start before you initialize the consumer. For details about initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

For further details about the options used in the command, enter:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt --help
```

3. Verify whether the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
init-status --suffix="dc=example,dc=com" example-agreement
Agreement successfully initialized.
```

Depending on the amount of data to replicate, the initialization can be time-consuming.

If you add multiple consumers for the suffix to the topology, repeat the steps on the supplier for each consumer. However, you must enable replication for the suffix only once on the supplier.

### 15.3.2. Setting up Single-supplier Replication Using the Web Console

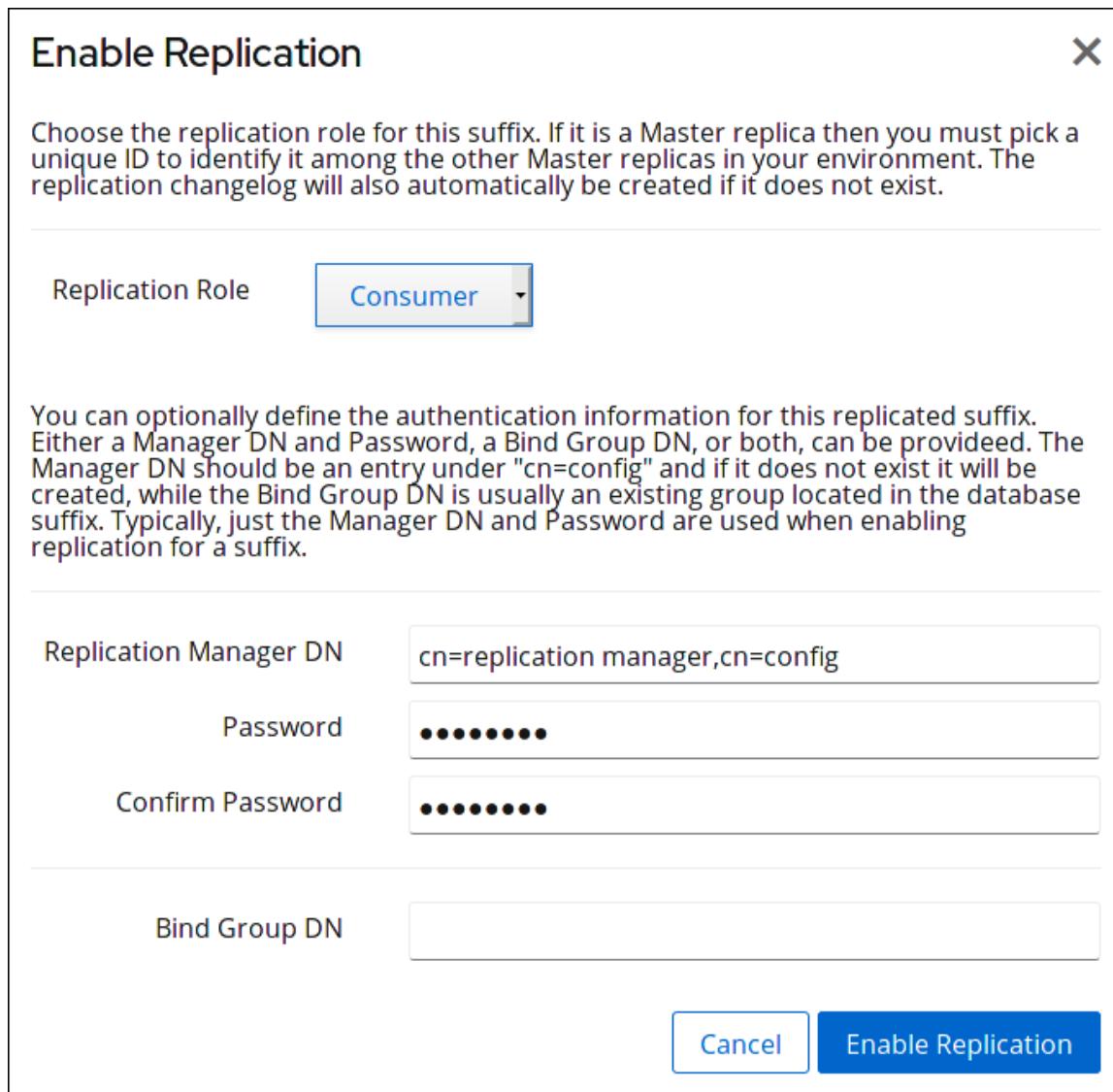
The following example assumes that you have an existing Directory Server instance running on a host named **supplier.example.com** that will act as a supplier in the replication topology to be set up. The following procedures describe how to add a read-only consumer named **consumer.example.com** to the topology, and how to configure single-supplier replication for the **dc=example,dc=com** suffix.

#### Steps to be Performed on the Consumer

On the **consumer.example.com** host:

1. Install Directory Server, and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. In case you created the instance without a database, create the database for the suffix. For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).
5. Enable replication for the suffix:

- a. Open the **Replication** menu.
- b. Select the **dc=example,dc=com** suffix, and click **Enable Replication**.
- c. Select **Consumer** in the **Replication Role** field, and enter the DN and password of the replication manager account to create. For example:



These settings configure the host as a consumer for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password, and allows this account to replicate changes for the suffix to this host.

- d. Click **Enable Replication**.

To add multiple consumers for the suffix to the topology, repeat the steps on each consumer.

### Steps to be Performed on the Supplier

On the **supplier.example.com** host:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Enable replication for the **dc=example,dc=com** suffix:

- a. Open the **Replication** menu.
- b. Select the **dc=example,dc=com** suffix, and click **Enable Replication**.
- c. Select **Supplier** in the **Replication Role** field, enter a replica ID, and leave the fields in the **Replication Authentication** area empty. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	<b>Supplier</b>
Replica ID	1

You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.

Replication Manager DN	
Password	
Confirm Password	
Bind Group DN	

This configures the host as a supplier for the **dc=example,dc=com** suffix, and sets the replica ID for this entry to **1**.



### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

- d. Click **Enable Replication**.
4. Add the replication agreement, and initialize the consumer:
  - a. Open the **Replication** menu, and select the **dc=example,dc=com** suffix.

- b. On Replication Agreements tab, click **Create Agreement**, and fill the fields. For example:

Field	Value
Agreement Name	example-agreement
Consumer Host	consumer.example.com
Consumer Port	636
Bind DN	cn=replication manager,cn=config
Bind Password	•••••••
Confirm Password	•••••••
Connection Protocol	LDAPS
Authentication Method	SIMPLE
Consumer Initialization	Do Online Initialization
<a href="#">Show Advanced Settings</a>	
<input type="button" value="Cancel"/> <input type="button" value="Save Agreement"/>	

These settings create a replication agreement named **example-agreement**. The replication agreement defines settings, such as the consumer's host name, protocol, and authentication information that the supplier uses when connecting and replicating data to the consumer.

- c. Select **Do Online Initialization** in the **Consumer Initialization** field to automatically initialize the consumer after saving the agreement.

To initialize the consumer later, select **Do Not Initialize**. Note that replication does not start before you initialize the consumer. For details about initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

- d. Click **Save Agreement**.

5. Verify whether the initialization was successful:

- a. Open the **Replication** menu.
- b. Select the **Agreements** entry.

For a successfully-completed initialization, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	Initialized

Depending on the amount of data to replicate, the initialization can be time-consuming.

If you add multiple consumers for the suffix to the topology, repeat the steps on the supplier for each consumer. However, you must enable replication for the suffix only once on the supplier.

## 15.4. MULTI-SUPPLIER REPLICATION

In a multi-supplier replication scenario, the supplier copies of the directory data are stored on multiple read-write replicas. Each of these servers maintains a changelog for the read-write replica. Directory Server supports up to 20 suppliers in a replication topology.



### NOTE

Each supplier in a multi-supplier replication environment is also a consumer automatically.

The following diagram shows a multi-supplier replication environment with two suppliers:

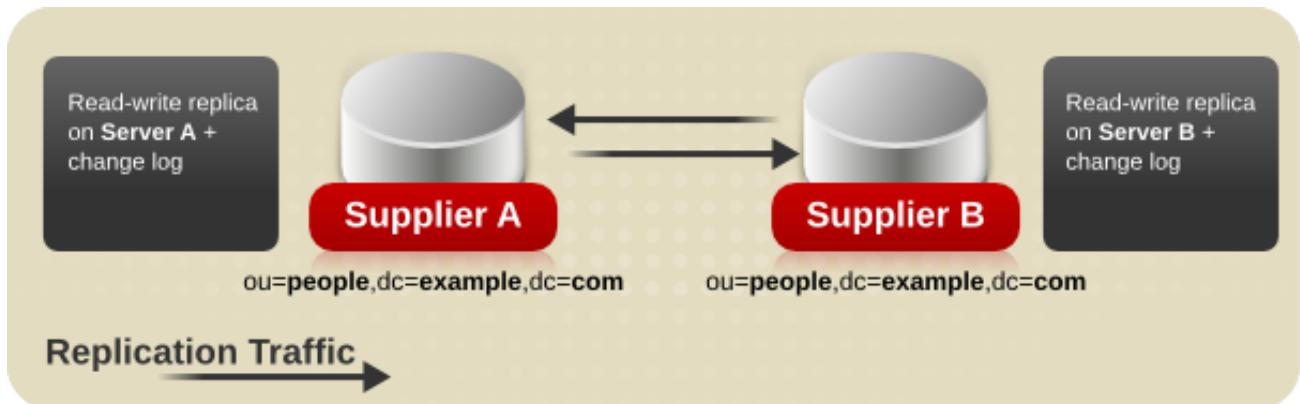


Figure 15.2. Multi-supplier Replication with Two Suppliers

In complex environments, replication topologies often contain multiple read-write suppliers as well as read-only consumers. The following diagram shows a topology where each supplier is configured with ten replication agreements to replicate data to two other suppliers and eight consumers:

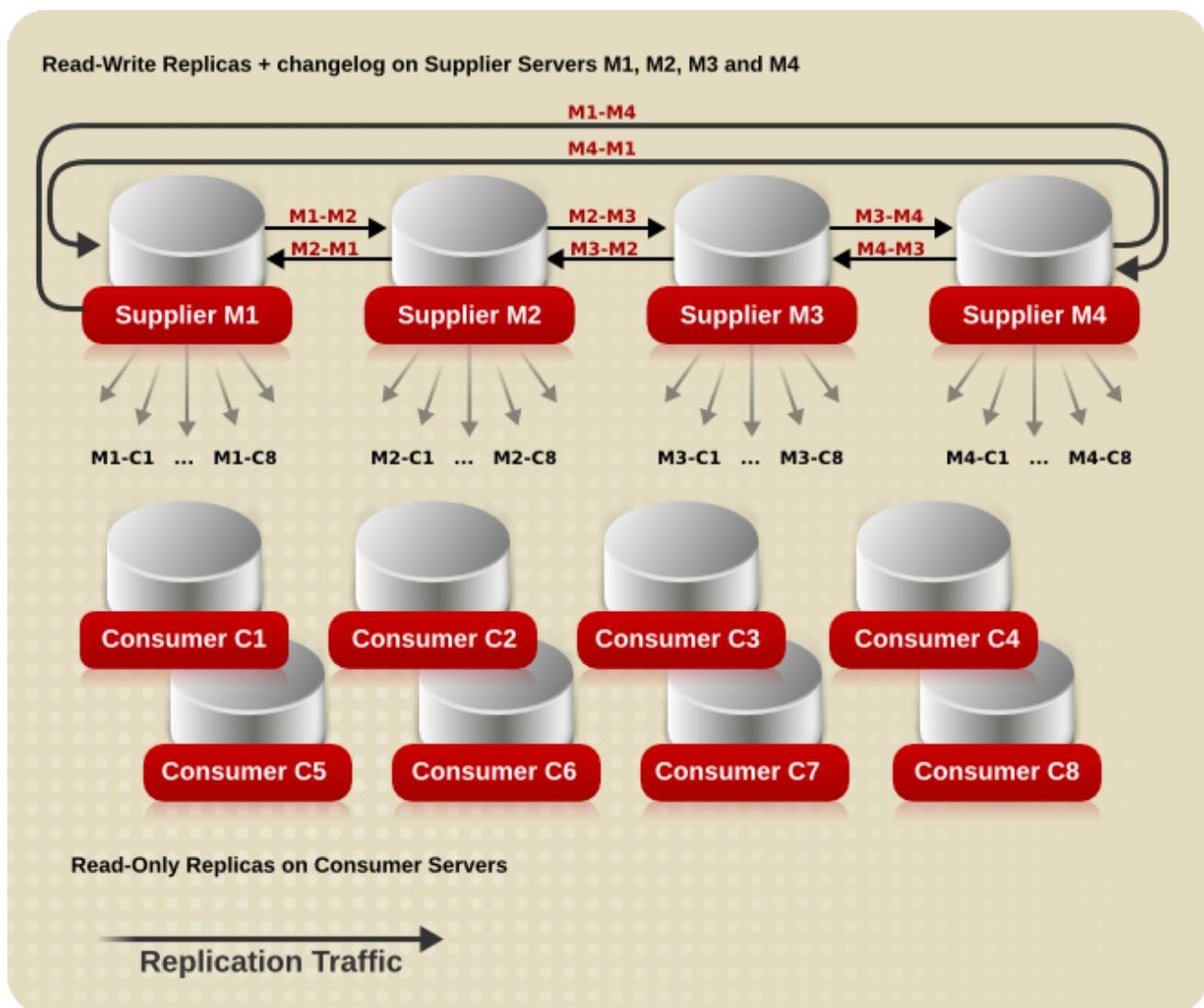


Figure 15.3. Complex Replication Scenario with Four Suppliers and Eight Consumers



### NOTE

The replication speed depends on:

- The speed of the network.
- The number of outgoing and incoming replication agreements.

Use the command line or web console to set up a multi-supplier replication topology. See:

- [Section 15.4.1, "Setting up Multi-supplier Replication Using the Command Line"](#)
- [Section 15.4.2, "Setting up Multi-supplier Replication Using the Web Console"](#)

### 15.4.1. Setting up Multi-supplier Replication Using the Command Line

The following example assumes that you have an existing Directory Server instance running on a host named **supplier1.example.com**. The following procedures describe how to add another read-write replica named **supplier2.example.com** to the topology, and how to configure multi-supplier replication for the **dc=example,dc=com** suffix.

#### Preparing the New Server to Join

On the **supplier2.example.com** host:

1. Install Directory Server, and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. In case you created the instance without a database, create the database for the suffix. For example, to create a database named **userRoot** for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://supplier2.example.com backend \
create --suffix="dc=example,dc=com" --be-name="userRoot"
```

For details on creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).

3. Enable replication for the suffix, and create the replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://supplier2.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id=1 \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```

This command configures the **supplier2.example.com** host as a supplier for the **dc=example,dc=com** suffix, and sets the replica ID for this entry to **1**. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password, and allows this account to replicate changes for the suffix to this host.



### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

## Configuring the Existing Server as a Supplier

On the **supplier1.example.com** host:

1. Similarly to the command you ran on the new server to join, enable replication for the **dc=example,dc=com** suffix, and create the replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id=2 \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```

The replica ID must be different than the one created in [the section called “Preparing the New Server to Join”](#), but the replication manager account can use the same DN.

2. Add the replication agreement, and initialize a new server. For example:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="supplier2.example.com" --port=636 \
--conn-protocol=LDAPS --bind-dn="cn=replication manager,cn=config" \
--bind-passwd="password" --bind-method=SIMPLE --init \
example-agreement-supplier1-to-supplier2
```

This command creates a replication agreement named **example-agreement-supplier1-to-supplier2**. The replication agreement defines settings, such as the consumer's host name, protocol, and authentication information that the supplier uses when connecting and replicating data to the consumer.

After the agreement was created, Directory Server initializes the consumer. To initialize the consumer later, omit the **--init** option. Note that replication does not start before you initialize the consumer. For details on initializing a consumer, see [Section 15.8.3, "Initializing a Consumer"](#).

For further details about the options used in the command, enter:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com repl-agmt --help
```

3. Verify whether the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com repl-agmt \
    init-status --suffix="dc=example,dc=com" example-agreement-supplier1-to-supplier2
Agreement successfully initialized.
```

Depending on the amount of data to replicate, the initialization can take be time-consuming.

### Configuring the New Server as a Supplier

On the **supplier2.example.com** host:



#### WARNING

Do not continue if you have not initialized the replication agreement on the existing server as described in [the section called "Configuring the Existing Server as a Supplier"](#). Otherwise, the empty database from the new server overrides the database on the existing supplier.

1. Add the replication agreement, and initialize the server. For example:

```
# dsconf -D "cn=Directory Manager" ldap://supplier2.example.com repl-agmt \
    create --suffix="dc=example,dc=com" --host="supplier1.example.com" --port=636 \
    --conn-protocol=LDAPS --bind-dn="cn=replication manager,cn=config" \
    --bind-passwd="password" --bind-method=SIMPLE --init \
    example-agreement-supplier2-to-supplier1
```

After the agreement was created, Directory Server initializes the consumer. To initialize the consumer later, omit the **--init** option.

2. Verify whether the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com repl-agmt \
    init-status --suffix="dc=example,dc=com" example-agreement-supplier2-to-supplier1
Agreement successfully initialized.
```

### 15.4.2. Setting up Multi-supplier Replication Using the Web Console

The following example assumes that you have an existing Directory Server instance running on a host named **supplier1.example.com**. The following procedures describe how to add another read-write replica named **supplier2.example.com** to the topology, and how to configure multi-supplier replication

for the **dc=example,dc=com** suffix.

## Preparing the New Server to Join

On the **supplier2.example.com** host:

1. Install Directory Server, and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. In case you created the instance without a database, create the database from the suffix. For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).
5. Enable replication for the suffix:
  - a. Open the **Replication** menu.
  - b. Select the **dc=example,dc=com** suffix, and click **Enable Replication**.
  - c. Select **Supplier** in the **Replication Role** field, enter a replica ID, as well as the DN and password of the replication manager account to create. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	<b>Supplier</b>
Replica ID	1
<p>You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.</p>	
Replication Manager DN	<code>cn=replication manager,cn=config</code>
Password	•••••••
Confirm Password	•••••••
Bind Group DN	
<input type="button" value="Cancel"/> <input type="button" value="Enable Replication"/>	

These settings configure the **supplier2.example.com** host as a supplier for the **dc=example,dc=com** suffix, and set the replica ID for this entry to **1**. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password, and allows this account to replicate changes for the suffix to this host.



### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

- d. Click **Enable Replication**.

#### Configuring the Existing Server as a Supplier

On the **supplier1.example.com** host:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.

3. Similarly to the settings on the new server to join, enable replication for the **dc=example,dc=com** suffix, and create a replication manager account:
  - a. Open the **Replication** menu.
  - b. Select the **dc=example,dc=com** suffix, and click **Enable Replication**.
  - c. Select **Supplier** in the **Replication Role** field, enter a replica ID, as well as the DN and password of the replication manager account to create. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	<input type="button" value="Supplier"/>
Replica ID	<input type="text" value="2"/> <input type="button" value=""/>
You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.	
Replication Manager DN	<input type="text" value="cn=replication manager,cn=config"/>
Password	<input type="password" value="*****"/>
Confirm Password	<input type="password" value="*****"/>
Bind Group DN	<input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Enable Replication"/>	

The replica ID must be different than the one created in [the section called “Preparing the New Server to Join”](#), but the replication manager account can use the same DN.

- d. Click **Enable Replication**.
4. Add the replication agreement and initialize the consumer:
  - a. Open the **Replication** menu, and select the **Agreements** entry.
  - b. Click **Create Replication Agreement**, and fill the fields. For example:

Create Replication Agreement X

Agreement Name	example-agreement-supplier1-to-supplier2
Consumer Host	supplier2.example.com
Consumer Port	636
Bind DN	cn=replication manager,cn=config
Bind Password	.....
Confirm Password	.....
Connection Protocol	LDAPS
Authentication Method	SIMPLE
Consumer Initialization	Do Online Initialization

[Show Advanced Settings](#)

Cancel Save Agreement

These settings create a replication agreement named **example-agreement-supplier1-to-supplier2**. The replication agreement defines settings, such as the consumer's host name, protocol, and authentication information that the supplier uses when connecting and replicating data to the consumer.

- Select **Do Online Initialization** in the **Consumer Initialization** field to automatically initialize the consumer after saving the agreement.

To initialize the consumer later, select **Do Not Initialize**. Note that replication does not start before you initialize the consumer. For details on initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

- Click **Save Agreement**.
- Verify whether the initialization was successful:
    - Open the **Replication** menu.
    - Select the **Agreements** entry.

For a successfully-completed initialization, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	<i>Initialized</i>

Depending on the amount of data to replicate, the initialization can be time-consuming.

### Configuring the New Server as a Supplier

On the **supplier2.example.com** host:



#### WARNING

Do not continue if you have not initialized the replication agreement on the existing server as described in [the section called “Configuring the Existing Server as a Supplier”](#). Otherwise, the empty database from the new server overrides the database on the existing supplier.

1. Add the replication agreement, and initialize the consumer:
  - a. Open the **Replication** menu, and select the **Agreements** entry.
  - b. Click **Create Replication Agreement**, and fill the fields. For example:

Create Replication Agreement X

Agreement Name	example-agreement-supplier2-to-supplier1
Consumer Host	supplier1.example.com
Consumer Port	636
Bind DN	cn=replication manager,cn=config
Bind Password	.....
Confirm Password	.....
Connection Protocol	LDAPS
Authentication Method	SIMPLE
Consumer Initialization	Do Online Initialization

[Show Advanced Settings](#)

Cancel Save Agreement

These settings create a replication agreement named **example-agreement-supplier2-to-supplier1**.

- Select **Do Online Initialization** in the **Consumer Initialization** field to automatically initialize the consumer after saving the agreement.

To initialize the consumer later, select **Do Not Initialize**. Note that replication does not start before you initialize the consumer. For details on initializing a consumer, see [Section 15.8.3, "Initializing a Consumer"](#).

- Click **Save Agreement**.
- Verify whether the initialization was successful:
    - Open the **Replication** menu.
    - Select the **Agreements** entry.

If the initialization completed successfully, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	Initialized

Depending on the amount of data to replicate, the initialization be time-consuming.

### 15.4.3. Preventing Monopolization of a Consumer in Multi-Supplier Replication

One of the features of multi-supplier 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. During a low update load, the supplier sends its update to another consumer while the first consumer is locked, and then sends updates when the first consumer is free again.

A problem can arise if the locking supplier is under a heavy update load or has a lot of pending updates in the changelog. If the locking supplier finishes sending updates and has multiple pending changes to send, it immediately attempts to reacquire the consumer. Such attempt in most cases succeeds, because other suppliers are usually sleeping. This can cause a single supplier to monopolize a consumer for several hours or longer.

The following attributes address this issue:

#### ***nsds5ReplicaBusyWaitTime***

Sets the time in seconds for a supplier to wait after a consumer sends back a busy response before making another attempt to acquire access.

For example, to configure that a supplier waits **5** seconds before making another acquire attempt:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
  set --suffix="suffix" --busy-wait-time=5 agreement_name
```

#### ***nsds5ReplicaSessionPauseTime***

Sets the time in seconds for a supplier to wait between two update sessions. If you set a value lower or equal than the value specified in ***nsds5ReplicaBusyWaitTime***, Directory Server automatically uses the value for the ***nsds5ReplicaSessionPauseTime*** parameter that is one second higher than the value set in ***nsds5ReplicaBusyWaitTime***.

For example, to configure that the supplier waits **10** seconds between two update sessions:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
  set --suffix="suffix" --session-pause-time=10 agreement_name
```

#### ***nsds5ReplicaReleaseTimeout***

Sets the timeout after which a supplier releases the replica, whether or not it has finished sending its updates. This prevents a single supplier from monopolizing a replica.

For example, to configure a supplier to release a replica after **90** seconds in a heavy replication environment:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication \
  set --suffix="suffix" --repl-release-timeout=90
```

For further details, see the parameter descriptions in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

To log replica busy errors, enable **Replication** error logging (log level **8192**). See [Section 21.3.7, “Configuring the Log Levels”](#).

## 15.5. CASCADING REPLICATION

In a cascading replication scenario, one server, a *hub*, acts both as a consumer and a supplier. It holds a read-only replica and maintains a changelog, so it receives updates from the supplier server that holds the supplier copy of the data and, in turn, supplies those updates to the consumer. Use cascading replication for balancing heavy traffic loads or to keep supplier servers based locally in geographically-distributed environments.

The following diagram shows a simple cascading replication scenario:

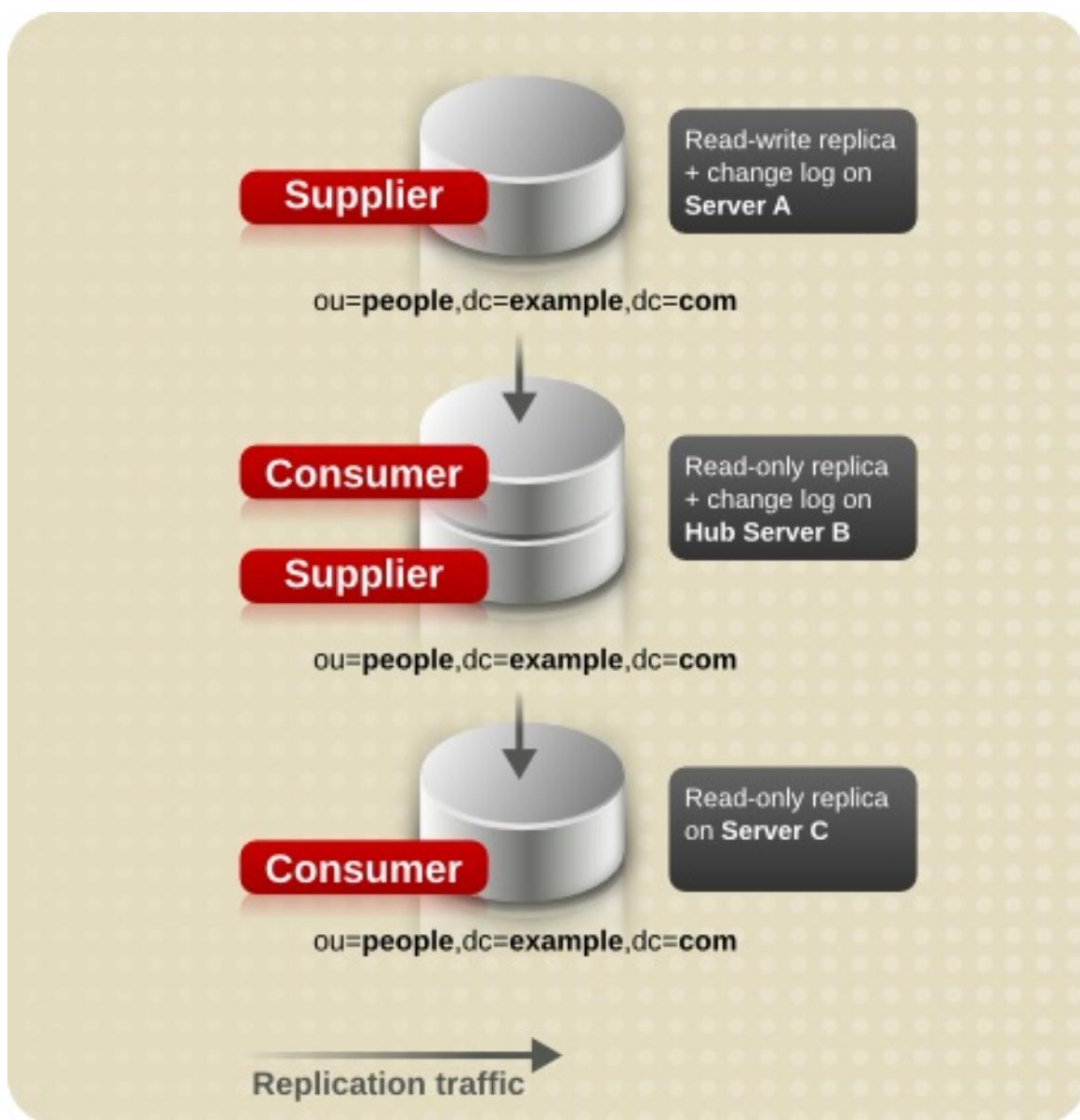


Figure 15.4. Cascading Replication

**NOTE**

Multi-supplier and cascading replication can be combined.

Use the command line or the web console to set up a cascading replication topology. See:

- [Section 15.5.1, “Setting up Cascading Replication Using the Command Line”](#)
- [Section 15.5.2, “Setting up Cascading Replication Using the Web Console”](#)

### 15.5.1. Setting up Cascading Replication Using the Command Line

The following example assumes that you have an existing Directory Server instance running on a host named **supplier.example.com**. The following procedures describe how to add a hub named **hub.example.com** to the topology that receives updates from the supplier for the **dc=example,dc=com** suffix. Subsequently, the procedure describes adding a consumer named **consumer.example.com** that receives updates from the hub server for the suffix.

#### Preparing the New Hub Server to Join

On the **hub.example.com** host:

1. Install Directory Server and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. In case you created the instance without a database, create the database for the suffix. For example, to create a database named **userRoot** for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://hub.example.com backend \
create --suffix="dc=example,dc=com" --be-name="userRoot"
```

For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).

3. Enable replication for the suffix and create the replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://hub.example.com replication \
enable --suffix="dc=example,dc=com" --role="hub" \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```

This command configures the **hub.example.com** host as a hub for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password and allows this account to replicate changes for the suffix to this host.

#### Configuring the Existing Server as a Supplier

On the **supplier.example.com** host:

1. Similarly to the command you ran on the new hub server to join in the section called “Preparing the New Hub Server to Join”, enable replication for the **dc=example,dc=com** suffix and create a replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://supplier1.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id=1 \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```



## IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

The replication manager account can use the same DN as the one created on the hub.

2. Add the replication agreement and initialize the hub. For example:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="hub.example.com" --port=636 \
--conn-protocol=LDAPS --bind-dn="cn=replication manager,cn=config" \
--bind-passwd="password" --bind-method=SIMPLE --init \
example-agreement-supplier-to-hub
```

This command creates a replication agreement named **example-agreement-supplier-to-hub**.

The replication agreement defines settings, such as the hubs' host name, protocol, and authentication information, which supplier uses when connecting and replicating data to the hub.

After the agreement was created, Directory Server initializes the hub. To initialize the hub later, omit the **--init** option. Note that replication does not start before you initialized the consumer. For details about initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

For further details about the options used in the command, enter:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt --help
```

3. Verify whether the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
init-status --suffix="dc=example,dc=com" example-agreement-supplier-to-hub
Agreement successfully initialized.
```

Depending on the amount of data to replicate, the initialization can be time-consuming.

## Preparing the new Consumer to Join

On the **consumer.example.com** host:

1. Install Directory Server and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. In case you created the instance without a database, create the database for the suffix. For example, to create a database named **userRoot** for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://hub.example.com backend \
create --suffix="dc=example,dc=com" --be-name="userRoot"
```

For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).

3. Enable replication for the suffix and create the replication manager account:

```
# dsconf -D "cn=Directory Manager" ldap://consumer.example.com replication \
enable --suffix="dc=example,dc=com" --role="consumer" \
--bind-dn="cn=replication manager,cn=config" --bind-passwd="password"
```

This command configures the **consumer.example.com** host as a consumer for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password and allows this account to replicate changes for the suffix to this host.

## Configuring the Hub as a Supplier for the Consumer

On the **hub.example.com** host:

1. Add the replication agreement and initialize the server. For example:

```
# dsconf -D "cn=Directory Manager" ldap://hub.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="consumer.example.com" --port=636 \
--conn-protocol=LDAPS --bind-dn="cn=replication manager,cn=config" \
--bind-passwd="password" --bind-method=SIMPLE --init \
example-agreement-hub-to-consumer
```

After the agreement was created, Directory Server initializes the consumer. To initialize the consumer later, omit the **--init** option.

2. Verify whether the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://hub.example.com repl-agmt \
init-status --suffix="dc=example,dc=com" example-agreement-hub-to-consumer
Agreement successfully initialized.
```

Depending on the amount of data to replicate, the initialization can be time-consuming.

## 15.5.2. Setting up Cascading Replication Using the Web Console

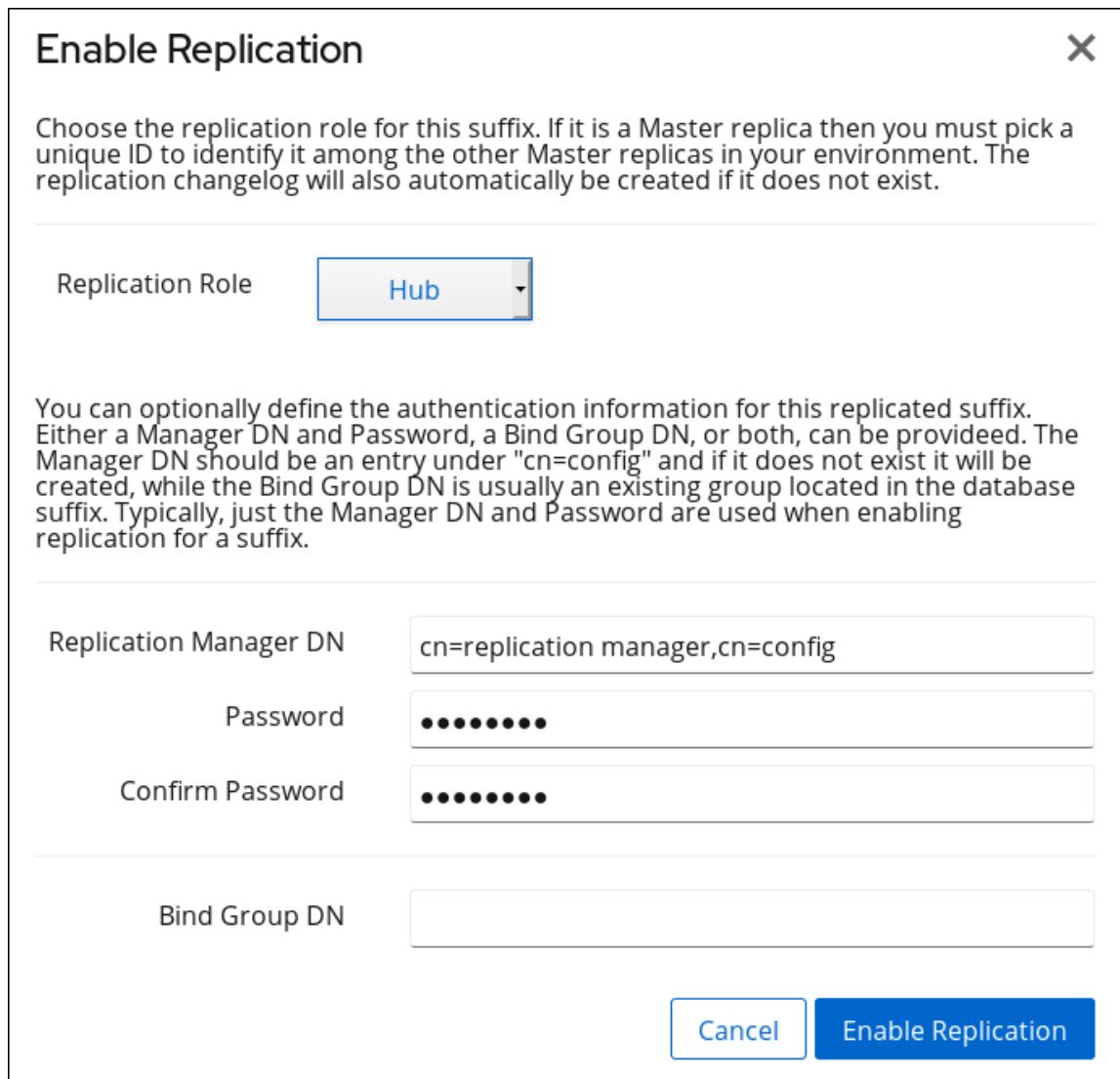
The following example assumes that you have an existing Directory Server instance running on a host named **supplier.example.com**. The following procedures describe how to add a hub named **hub.example.com** to the topology that receives updates from the supplier for the **dc=example,dc=com** suffix. Subsequently, the procedure describes adding a consumer named **consumer.example.com** that receives updates from the hub server for the suffix.

### Preparing the New Hub Server to Join

On the **hub.example.com** host:

1. Install Directory Server and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. In case you created the instance without a database, create the database for the suffix. For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).
5. Enable replication for the suffix:

- a. Open the **Replication** menu.
- b. Select the **dc=example,dc=com** suffix and click **Enable Replication**.
- c. Select **Hub** in the **Replication Role** field, and enter the DN and password of the replication manager account to create. For example:



These settings configure the **hub.example.com** host as a hub for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password and allows this account to replicate changes for the suffix to this host.

- d. Click **Enable Replication**.

#### Configuring the Existing Server as a Supplier On the **supplier.example.com** host:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Similarly to the settings on the new hub server to join in [the section called “Preparing the New Hub Server to Join”](#), enable replication for the **dc=example,dc=com** suffix and create a replication manager account:

- a. Open the **Replication** menu.
- b. Select the **dc=example,dc=com** suffix and click **Enable Replication**.
- c. Select **Supplier** in the **Replication Role** field, enter a replica ID, as well as the DN and password of the replication manager account to create. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	<input type="button" value="Supplier"/>
Replica ID	<input type="text" value="1"/> <input type="button" value=""/>
You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.	
Replication Manager DN	<input type="text" value="cn=replication manager,cn=config"/>
Password	<input type="password" value="*****"/>
Confirm Password	<input type="password" value="*****"/>
Bind Group DN	<input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Enable Replication"/>	



### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

- The replication manager account can use the same DN as the one created on the hub.
- d. Click **Enable Replication**.
  4. Add the replication agreement and initialize the hub:
    - a. Open the **Replication** menu, and select the suffix.

- b. On the **Replication Agreements** tab, click **Create Agreement**, and fill the fields. For example:

Field	Value
Agreement Name	example-agreement-supplier-to-hub
Consumer Host	hub.example.com
Consumer Port	636
Bind DN	cn=replication manager,cn=config
Bind Password	.....
Confirm Password	.....
Connection Protocol	LDAPS
Authentication Method	SIMPLE
Consumer Initialization	Do Online Initialization

[Show Advanced Settings](#)

[Cancel](#) [Save Agreement](#)

These settings create a replication agreement named **example-agreement-supplier-to-hub**. The replication agreement defines settings, such as the hubs' host name, protocol, and authentication information, the supplier uses when connecting and replicating data to the hub.

- c. Select **Do Online Initialization** in the **Consumer Initialization** field to automatically initialize the consumer after saving the agreement.

To initialize the hub later, select **Do Not Initialize**. Note that replication does not start before you initialized the consumer. For details about initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

- d. Click **Save Agreement**.

5. Verify whether the initialization was successful:

- Open the **Replication** menu.
- Select the **Agreements** entry.

If the initialization completed successfully, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	<i>Initialized</i>

Depending on the amount of data to replicate, the initialization can be time-consuming.

### Configuring the New Consumer to Join

On the **consumer.example.com** host:

1. Install Directory Server and create an instance. For details, see the [Red Hat Directory Server Installation Guide](#).
2. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
3. Select the instance.
4. In case you created the instance without a database, create the database for the suffix. For details about creating a database for a suffix, see [Section 2.1.1, “Creating Suffixes”](#).
5. Enable replication for the suffix:
  - a. Open the **Replication** menu.
  - b. Select the **dc=example,dc=com** suffix and click **Enable Replication**.
  - c. Select **Consumer** in the **Replication Role** field, and enter the DN and password of the replication manager account to create. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	Consumer
You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.	
Replication Manager DN	<code>cn=replication manager,cn=config</code>
Password	••••••••
Confirm Password	••••••••
Bind Group DN	
<input type="button" value="Cancel"/> <input type="button" value="Enable Replication"/>	

These settings configure the **consumer.example.com** host as a consumer for the **dc=example,dc=com** suffix. Additionally, the server creates the **cn=replication manager,cn=config** user with the specified password and allows this account to replicate changes for the suffix to this host.

- d. Click **Enable Replication**.

#### Configuring the Hub as a Supplier for the Consumer

On the **consumer.example.com** host:

1. Add the replication agreement and initialize the consumer:
  - a. Open the **Replication** menu, and select the suffix.
  - b. On the **Replication Agreements** tab, click **Create Agreement**, and fill the fields. For example:

**Create Replication Agreement**

Agreement Name	example-agreement-hub-to-consumer
Consumer Host	hub.example.com
Consumer Port	636
Bind DN	cn=replication manager,cn=config
Bind Password	*****
Confirm Password	*****
Connection Protocol	LDAPS
Authentication Method	SIMPLE
Consumer Initialization	Do Online Initialization
<a href="#">Show Advanced Settings</a>	
<input type="button" value="Cancel"/> <input type="button" value="Save Agreement"/>	

These settings create a replication agreement named **example-agreement-hub-to-consumer**.

- c. Select **Do Online Initialization** in the **Consumer Initialization** field to automatically initialize the consumer after saving the agreement.

To initialize the consumer later, select **Do Not Initialize**. Note that replication does not start before you initialized the consumer. For details about initializing a consumer, see [Section 15.8.3, “Initializing a Consumer”](#).

- d. Click **Save Agreement**.
2. Verify whether the initialization was successful:
  - a. Open the **Replication** menu.
  - b. Select the **Agreements** entry.

If the initialization completed successfully, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	Initialized

Depending on the amount of data to replicate, the initialization can be time-consuming.

## 15.6. CONFIGURING REPLICATION PARTNERS TO USE CERTIFICATE-BASED AUTHENTICATION

Instead of using a bind DN and password to authenticate to a replication partner, you can use certificate-based authentication.

The following procedure describes how to add a new server named **server2.example.com** to the replication topology, and how to set up replication agreements between the new host and the existing **server1.example.com** using certificate-based authentication:

1. On both hosts, set up certificate-based authentication. For details, see [Section 9.9.1, “Setting up Certificate-based Authentication”](#).
2. On the **server1.example.com** host:
  - a. Create accounts for both servers, such as **cn=server1,example,dc=com** and **cn=server2,dc=example,dc=com** and add the client certificates to the corresponding accounts. For details, see:
    - [Section 3.3.1, “Adding an Entry Using \*\*Idapadd\*\*”](#)
    - [Section 9.9.2, “Adding a Certificate to a User”](#)

Both servers will later use these accounts and certificates to authenticate when they establish a replication connection to each other.

- b. Create a group, such as **cn=repl\_server,ou=Groups,dc=example,dc=com**, and add both server accounts. See [Section 8.1, “Using Groups”](#).
- c. Create the replica entry and set the **nsds5ReplicaBindDNGroup** attribute to the DN of the group created in the previous step:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id="7" \
--bind-group-dn="cn=repl_server,ou=Groups,dc=example,dc=com"
```

- d. Set the replica entry's interval in which Directory Server checks if the group has been changed to **0**:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com replication \
set --suffix="dc=example,dc=com" --repl-bind-group-interval=0
```

3. Initialize the new server:

- a. Create a temporary replication manager account, such as **cn=Replication Manager,cn=config**, on **server2.example.com**.
- b. On **server1.example.com**, create a temporary replication agreement which uses the account from the previous step for authentication:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="server1.example.com" --port=636 \
--conn-protocol=LDAPS --bind-dn="cn=Replication Manager,cn=config" \
--bind-passwd="password" --bind-method=SIMPLE --init \
temporary_agreement
```

This agreement uses the previously-created replication manager account to initialize the database. Before this initialization, the database on **server2.example.com** is empty and the accounts with the associated certificates do not exist. Therefore, replication using certificates is not possible before the database is initialized.

4. After the new server has been initialized:

- a. Remove the temporary replication agreement from **server1.example.com**:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com repl-agmt \
delete --suffix="dc=example,dc=com" temporary_agreement
```

- b. Remove the temporary replication manager account from **server2.example.com**:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com replication \
delete-manager --suffix="dc=example,dc=com" --name="Replication Manager"
```

5. Create a replication agreement on both servers that use certificate-based authentication:

- a. On **server1.example.com**:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="server2.example.com" --port=636 \
--conn-protocol=LDAPS --bind-method="SSLCLIENTAUTH" \
--init example_agreement
```

- b. On **server2.example.com**:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com repl-agmt \
create --suffix="dc=example,dc=com" --host="server1.example.com" --port=636 \
--conn-protocol=LDAPS --bind-method="SSLCLIENTAUTH" \
--init example_agreement
```

6. To verify the replication works correctly, display the **nsds5replicaLastUpdateStatus** attribute in the replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com repl-agmt status -- \
suffix="dc=example,dc=com" example_agreement
```

For details about possible statuses, see the [Replication Agreement Status](#) appendix in the *Red Hat Directory Server Configuration, Command, and File Reference*.

## 15.7. PROMOTING A CONSUMER OR HUB TO A SUPPLIER

In certain situations, such as when a supplier in a replication topology is unavailable due to a hardware outage, administrators want to promote a read-only consumer or hub to a writable supplier.

### 15.7.1. Promoting a Consumer or Hub to a Supplier Using the Command Line

For example, to promote the **server.example.com** host to a supplier for the **dc=example,dc=com** suffix:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication \
promote --suffix="dc=example,dc=com" --newrole="supplier" --replica-id=2
```



#### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

Optionally, you can now configure the new supplier to replicate changes for the suffix to other servers in the topology. For details about configuring replication, see:

- [Section 15.3.1, “Setting up Single-supplier Replication Using the Command Line”](#)
- [Section 15.4.1, “Setting up Multi-supplier Replication Using the Command Line”](#)
- [Section 15.5.1, “Setting up Cascading Replication Using the Command Line”](#)

### 15.7.2. Promoting a Consumer or Hub to a Supplier Using the Web Console

For example, to promote a consumer or hub to a supplier for the **dc=example,dc=com** suffix:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Replication** menu and select the **Configuration** entry.
4. Select the **dc=example,dc=com** suffix.
5. Click **Promote**.
6. Select **Supplier** in the **Replication Role** field and enter a replica ID.



#### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

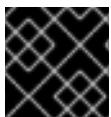
7. Select **Yes, I am sure**.
8. Click **Change Role** to confirm the new role.

Optionally, you can now configure the new supplier to replicate changes for the suffix to other servers in the topology. For details about configuring replication, see:

- [Section 15.3.2, “Setting up Single-supplier Replication Using the Web Console”](#)
- [Section 15.4.2, “Setting up Multi-supplier Replication Using the Web Console”](#)
- [Section 15.5.2, “Setting up Cascading Replication Using the Web Console”](#)

## 15.8. ABOUT INITIALIZING A CONSUMER

After creating the replication agreement, initialize the consumer. During this operation, the supplier copies the existing data to the consumer.



### IMPORTANT

Replication will not begin until you initialized the consumer.

#### 15.8.1. When to Initialize a Consumer

Consumer initialization involves copying data from the supplier server to the consumer server. Once the subtree has been physically placed on the consumer, the supplier server can begin replaying update operations to the consumer server.

Under normal operations, the consumer should not ever have to be reinitialized. However, any time there is a chance that there is a big discrepancy between the supplier's data and the consumer's, reinitialize the consumer. For example, if the data on the supplier server is restored from backup, then all consumers supplied by that server should be reinitialized. As another example, if the supplier has not been able to contact the consumer for a long time, like a week, the supplier may determine that the consumer is too far out of date to be updated, and must be reinitialized.

The consumer can either be initialized online using the web console or manually using the command line. Online consumer initialization using the web console is an effective method of initializing a small number of consumers. However, since each replica is initialized in sequence, this method is not suited to initializing a large number of replicas. Online consumer initialization is the method to use when the consumer is initialized as part of configuring the replication agreement on the supplier server.

Manual consumer initialization using the command line is a more effective method of initializing a large number of consumers from a single LDIF file.

#### 15.8.2. Setting Initialization Timeouts

If initialization of large databases fails due to timeouts, set one of the following to a large enough time period or to an unlimited period to enable Directory Server to initialize the entire database before the operation times out:

- The **nsslapd-idletimeout** configuration parameter in the **cn=config** entry sets the timeout for all replication agreements on the server. For example, to disable the timeout globally:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-idletimeout=0
```

- The ***nsIdleTimeout*** parameter in the replication manager's DN set's the timeout for all agreements that use this replication manager entry. For example, to disable the timeout for the **cn=replication manager,cn=config** entry:

```
# ldapmodify -D "cn=Directory Manager" -w -h server.example.com -p 389 -x
dn: cn=replication manager,cn=config
changetype: modify
add: nsIdleTimeout
nsIdleTimeout: 0
```

### 15.8.3. Initializing a Consumer

This section describes initializing a consumer using the command line and in the web console.

#### 15.8.3.1. Initializing a Consumer Using the Command Line

You can initialize a consumer online and offline using the command line. This section explains both procedures.

##### 15.8.3.1.1. Initializing a Consumer Online

After creating the replication agreement, use the **dsconf repl-agmt init** command to initialize a consumer online:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt \
init --suffix="suffix" agreement_name
```

##### 15.8.3.1.2. Initializing a Consumer Offline

To initialize a consumer offline:

1. On the supplier:

- a. Shutdown the instance on the supplier:

```
# dsctl instance_name stop
```

- b. Export the **userRoot** database that contains the suffix to replicate into the **/tmp/example.ldif** file with replication information:

```
# dsctl instance_name db2ldif --replication userRoot /tmp/example.ldif
```

- c. Start the instance on the supplier:

```
# dsctl instance_name start
```

2. Copy the exported file to the consumer.
3. Import the data on the consumer. For details, see [Section 6.1.2.2, “Importing Data While the Server is Offline”](#).

#### 15.8.4. Initializing a Consumer Using the Web Console

To initialize a consumer online using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Replication** menu, and select the suffix.
4. On the **Replication Agreements** tab, open the **Choose Action** menu next to the replication agreement for the suffix and select **Initialize Agreement**.

If the initialization completed successfully, web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
<b>Error (0) Replica acquired successfully: Incremental update succeeded</b>	<b>Initialized</b>

Depending on the amount of data to replicate, the initialization can be time-consuming.

## 15.9. DISABLING AND RE-ENABLING REPLICATION

By default, replication is enabled when you create a replication agreement. In certain situations, for example, when taking down a server for maintenance, administrators want to temporarily disable replication.



### IMPORTANT

If you disable the replication entry using the **dsconf replication disable** command, Directory Server automatically deletes the replication agreement as well. In this case, to re-enable replication, you must recreate the replication agreement.

To temporarily disable a replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt disable --suffix="dc=example,dc=com" agreement_name
```

To re-enable an replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt enable --suffix="dc=example,dc=com" agreement_name
```

## 15.10. REMOVING A DIRECTORY SERVER INSTANCE FROM THE REPLICATION TOPOLOGY

In certain situations, such as hardware outages or structural changes, administrators want to remove Directory Server instances from a replication topology. This section explains the details about removing an instance.

### 15.10.1. Removing a Consumer or Hub from the Replication Topology

To remove a consumer or hub from the replication topology:

1. If the host to remove is a hub and also a supplier for other servers in the topology, configure other suppliers or hubs to replicate data to these servers. If these servers have no other supplier configured and you remove the hub, these servers become isolated from the replication topology. For details about configuring replication, see:
  - [Section 15.3, "Single-supplier Replication"](#)
  - [Section 15.4, "Multi-Supplier Replication"](#)
  - [Section 15.5, "Cascading Replication"](#)
2. On the host to remove, set the database into read-only mode to prevent any updates:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h host-to-remove.example.com -x

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

3. On all suppliers that have a replication agreement with the host to remove, delete the replication agreements. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt \
  delete --suffix="dc=example,dc=com" agreement_name
```

4. On the consumer or hub to remove disable replication for all suffixes. For example:

```
# dsconf -D "cn=Directory Manager" ldap://host-to-remove.example.com replication \
  disable --suffix="dc=example,dc=com"
```

Disabling replication automatically deletes all replication agreements for this suffix on this server.

### 15.10.2. Removing a Supplier from the Replication Topology

Removing a supplier cleanly from the replication topology is more complex than removing a consumer or hub. This is because every supplier in the topology stores information about other suppliers, and they retain that information even if a supplier suddenly becomes unavailable.

Directory Server maintains information about the replication topology in a set of meta data called the replica update vector (RUV). The RUV contains information about the supplier, such as its ID, URL, latest change state number (CSN) on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

To remove a supplier cleanly, you must remove its meta data along with the configuration entries.

1. If the replica to be removed is also a supplier for other servers in the topology, configure other suppliers or hubs to replicate data to these servers. If these servers have no other supplier configured and you remove the supplier, these servers become isolated from the replication topology. For details about configuring replication, see:

- [Section 15.3, "Single-supplier Replication"](#)
- [Section 15.4, "Multi-Supplier Replication"](#)
- [Section 15.5, "Cascading Replication"](#)

2. On the supplier to remove:

- a. Set the database into read-only mode to prevent any updates. For details, see [Section 2.2.2.1, "Setting a Database in Read-Only Mode"](#).
- b. Wait until all other servers in the topology received all data from this supplier. To verify, ensure that the CSN on other servers is equal or greater than the CSN on the supplier to remove. For example:

```
# ds-replcheck online -D "cn=Directory Manager" -w password -m ldap://replica-to-remove.example.com:389 -r ldap://server.example.com:389 -b dc=example,dc=com
=====
=====
Replication Synchronization Report (Tue Mar 5 09:46:20 2019)
=====
=====

Database RUV's
=====

Supplier RUV:
{replica 1 ldap://replica-to-remove.example.com:389} 5c7e8927000100010000
5c7e89a0000100010000
{replicageneration} 5c7e8927000000010000

Replica RUV:
{replica 1 ldap://replica-to-remove.example.com:389} 5c7e8927000100010000
5c7e8927000400010000
{replica 2 ldap://server.example.com:389}
{replicageneration} 5c7e8927000000010000
```

- c. Display the replica ID:

```
# dsconf -D "cn=Directory Manager" ldap://replica-to-remove.example.com replication get
--suffix="dc=example,dc=com" | grep -i "nsds5replicaid"
nsDS5Replicaid: 1
```

In this example, the replica ID is **1**. Remember your replica ID for the last step of this procedure.

3. On all suppliers that have a replication agreement with the replica to remove, delete the replication agreements. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt \
    delete --suffix="dc=example,dc=com" agreement_name
```

- On the replica to remove, disable replication for all suffixes. For example:

```
# dsconf -D "cn=Directory Manager" ldap://replica-to-remove.example.com replication \
    disable --suffix="dc=example,dc=com"
```

Disabling replication automatically deletes all replication agreements for this suffix on this server.

- On one of the remaining suppliers in the topology, clean the RUVs for the replica ID. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-tasks \
    cleanallruv --suffix="dc=example,dc=com" --replica-id=1
```

The command requires to specify the replica ID displayed in an earlier step of this procedure.

## 15.11. MANAGING ATTRIBUTES WITHIN FRACTIONAL REPLICATION

As [Section 15.1.7, “Replicating a Subset of Attributes with Fractional Replication”](#) describes, fractional replication allows administrators to set attributes that are *excluded* from replication updates. Administrators can do this for a variety of performance reasons – to limit the number of large attributes that are sent over a network or to reduce the number of times that fixup tasks (like **memberOf** calculations) are run.

The list of attributes to exclude from replication are defined in the ***nsDS5ReplicatedAttributeList*** attribute. This attribute is part of the replication agreement and it can be configured in the replication agreement wizard in the web console or through the command line when the replication agreement is created.

```
nsDS5ReplicatedAttributeList: (objectclass=*) $ EXCLUDE memberof authorityRevocationList
accountUnlockTime
```



### IMPORTANT

Directory Server requires the **(objectclass=\*) \$ EXCLUDE** part in the value of the ***nsDS5ReplicatedAttributeList*** attribute. If you edit the attribute directly, for example using the **ldapmodify** utility, you must specify this part together with the list of attributes as displayed in the example above. However, both the **dsconf** and web console automatically add the **(objectclass=\*) \$ EXCLUDE** part, and you must only specify the attributes.

#### 15.11.1. Setting Different Fractional Replication Attributes for Total and Incremental Updates

When fractional replication is first configured, the list of excluded attributes applies to every update operation. Meaning, this list of attributes is excluded for a total update as well as regular incremental updates. However, there can be times when attributes should be excluded from incremental updates for performance but should be included in a total update to ensure the directory data sets are complete. In this case, it is possible to add a second attribute that defines a separate list of attributes to exclude from total updates, ***nsDS5ReplicatedAttributeListTotal***.



## NOTE

***nsDS5ReplicatedAttributeList*** is the primary fractional replication attribute. If only ***nsDS5ReplicatedAttributeList*** is set, then it applies to both incremental updates and total updates. If both ***nsDS5ReplicatedAttributeList*** and ***nsDS5ReplicatedAttributeListTotal*** are set, then ***nsDS5ReplicatedAttributeList*** only applies to incremental updates.

For example, every time a ***memberOf*** attribute is added to an entry, a ***memberOf*** fixup task is run to resolve the group membership. This can cause overhead on the server if that task is run every time replication occurs. Since a total update only occurs for a database which is newly-added to replication or that has been offline for a long time, running a ***memberOf*** fixup task after a total update is an acceptable option. In this case, the ***nsDS5ReplicatedAttributeList*** attribute lists ***memberOf*** so it is excluded from incremental updates, but ***nsDS5ReplicatedAttributeListTotal*** does not list ***memberOf*** so that it is included in total updates.

The exclusion list for incremental updates is set in the ***nsDS5ReplicatedAttributeList*** attribute for the replication agreement. For example:

```
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList accountUnlockTime
memberof
```

To set the ***nsDS5ReplicatedAttributeList*** attribute, use the **dsconf repl-agmt set** command. For example:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt set \
--suffix="suffix" --frac-list="authorityRevocationList accountUnlockTime memberof" \
agreement_name
```

If ***nsDS5ReplicatedAttributeList*** is the only attribute set, then that list applies to both incremental and total updates. To set a separate list for total updates, add the ***nsDS5ReplicatedAttributeListTotal*** attribute to the replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt set \
--suffix="suffix" --frac-list-total="accountUnlockTime" \
agreement_name
```



## NOTE

The ***nsDS5ReplicatedAttributeList*** attribute must be set for incremental updates before ***nsDS5ReplicatedAttributeListTotal*** can be set for total updates.

### 15.11.2. The Replication Keep-alive Entry

When you update an attribute on a supplier, the changelog change sequence number (CSN) is increased on the supplier. In a replication topology, this server now connects to the first consumer and compares the local CSN with the CSN on the consumer. If it is lower, the update is retrieved from the local changelog and replicated to the consumer. In a replication topology with fractional replication enabled, this can cause problems: For example, if only attributes are updated on the supplier that are excluded from replication, no update to replicate is found, and therefore the CSN is not updated on the consumer. In certain scenarios, such as when only attributes are updated on a supplier that are excluded from replication, unnecessary searching for updates on the supplier can cause other servers to receive the data later than needed. To work around this problem, Directory Server uses keep-alive entries.

If all updated attributes on the supplier are excluded from replication and the number of skipped updates exceeds 100, the ***keepalivetimestamp*** attribute is updated on the supplier and replicated to the consumer. Because the ***keepalivetimestamp*** attribute is not excluded from replication, the update of the keep-alive entry is replicated, the CSN on the consumer is updated, and then equal to the one on the supplier. The next time the supplier connects to the consumer, only updates that are newer than the CSN on the consumer are searched. This reduces the amount of time spent by a supplier to search for new updates to send.

Directory Server automatically creates the replication keep-alive entry on demand on a supplier. It contains the replica ID of the supplier in the distinguished name (DN). Each keep-alive entry is specific to a given supplier. For example, to display the hidden keep-alive entry:

```
# ldapsearch -D "cn=Directory Manager" -b "dc=example,dc=com" -W -p 389 -h server.example.com
-x 'objectClass=ldapsubentry'

dn: cn=repl keep alive 1,dc=example,dc=com
objectclass: top
objectclass: ldapsubentry
objectclass: extensibleObject
cn: repl keep alive 1
keepalivetimestamp: 20181112150654Z
```

The keep-alive entry is updated in the following situations (if it does not exist before the update, it is created first):

- When a fractional replication agreement skips more than 100 updates and does not send any updates before ending the replication session.
- When a supplier initializes a consumer, initially it creates its own keep-alive entry. A consumer that is also a supplier does not create its own keep-alive entry unless it also initializes another consumer.

### 15.11.3. Preventing "Empty" Updates from Fractional Replication

Fractional replication allows a list of attributes which are removed from replication updates (***nsDS5ReplicatedAttributeList***). However, a changed to an excluded attribute still triggers a modify event and generates an empty replication update.

The ***nsds5ReplicaStripAttrs*** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attributes like ***modifiersName***.

For example, let's say that the ***accountUnlockTime*** attribute is excluded. John Smith's user account is locked and then the time period expires and it is automatically unlocked. Only the ***accountUnlockTime*** attribute has changed, and that attribute is excluded from replication. However, the operational attribute ***internalmodifytimestamp*** also changed. A replication event is triggered because John Smith's user account was modified – but the only data to send is the new modify time stamp and the update is otherwise empty. If there are a large number of attributes related to login times or password expiration times (for example), this could create a flood of empty replication updates that negatively affect server performance or that interfere with associated applications.

To prevent this, add the ***nsds5ReplicaStripAttrs*** attribute to the replication agreement to help tune the fractional replication behavior:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com repl-agmt set \
```

```
--suffix="suffix" \
--strip-list="modifiersname modifytimestamp internalmodifiersname" \
agreement_name
```

If a replication event is *not* empty, the stripped attributes are still replicated with the other changes. These attributes are removed from updates only if the event would otherwise be empty.

## 15.12. MANAGING DELETED ENTRIES WITH REPLICATION

When an entry is deleted, it is not immediately removed from the database. Rather, it is converted into a *tombstone* entry, a kind of backup entry that is used by servers in replication to resolve specific conflicts (orphaned entries). The tombstone entry is the original entry with a modified DN, an added **nsTombstone** object class, but the attributes are removed from the index.

Tombstones are not preserved indefinitely. A purge job is run periodically, at a specified interval (set in the **nsDS5ReplicaTombstonePurgeInterval** attribute); the purge removes old tombstone entries. Tombstone entries are saved for a given amount of time (set in the **nsDS5ReplicaPurgeDelay** attribute); once a tombstone entry is older than the delay period, it is reaped at the next purge job.

Both the purge delay and the purge interval are set on the replica entry in the **cn=replica,cn=replicated suffix,cn=mapping tree,cn=config** configuration entry. There are two considerations when defining the purge settings for replication:

- The purge operation is time-consuming, especially if the server handles a lot of delete operations. Do not set the purge interval too low or it could consume too many server resources and affect performance.
- Suppliers use change information, including tombstone entries, to prime replication after initialization. There should be enough of a backlog of changes to effectively re-initialize consumers and to resolve replication conflicts. Do not set the purge delay (the age of tombstone entries) too low or you could lose information required to resolve replication conflicts.

Set the purge delay so that it is slightly longer than the longest replication schedule in the replication topology. For example, if the longest replication interval is 24 hours, keep tombstone entries around for 25 hours. This ensures that there is enough change history to initialize consumers and prevent the data stored in different suppliers from diverging.

When you use the **dsconf replication set** command, the **--repl-tombstone-purge-interval=seconds** option sets the **nsDS5ReplicaTombstonePurgeInterval** attribute and the **--repl-purge-delay=seconds** option the **nsDS5ReplicaPurgeDelay** attribute.

For example, to set the tombstone purge interval to **43200** (12 hours) and the replica purge delay to **90000** (25 hours):

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication set \
--repl-tombstone-purge-interval=43200 --repl-purge-delay=90000
```



### NOTE

To clean up the tombstone entries and the state information immediately, set a very small value to the **nsDS5ReplicaTombstonePurgeInterval** and **nsDS5ReplicaPurgeDelay** attributes. Both attributes have values set in seconds, so the purge operations can be initiated almost immediately.

**WARNING**

Always use the purge intervals to clean out tombstone entries from the database.  
**Never delete tombstone entries manually.**

## 15.13. CONFIGURING CHANGELOG ENCRYPTION

To increase security, Directory Server supports encrypting the changelog. This section explains how to enable this feature.

### Prerequisites

The server must have a certificate and key stored in the network security services (NSS) database. Therefor, enable TLS encryption on the server as described in [Section 9.4.1, “Enabling TLS in Directory Server”](#).

### Procedure

To enable changelog encryption:

1. Except for the server on which you want to enable changelog encryption, stop all instances in the replication topology by entering the following command:

```
# dsctl instance_name stop
```

2. On the server where you want to enable changelog encryption:

- a. Export the changelog, for example, to the **/tmp/changelog.ldif** file:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication dump-changelog -o /tmp/changelog.ldif
```

- b. Stop the instance:

```
# dsctl instance_name stop
```

- c. Add the following setting to the **dn: cn=changelog5,cn=config** entry in the **/etc/dirsrv/slappd-*instance\_name*/dse.ldif** file:

```
nsslapd-encryptionalgorithm: AES
```

- d. Start the instance:

```
# dsctl instance_name start
```

- e. Import the changelog from the **/tmp/changelog.ldif** file:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication restore-changelog from-ldif /tmp/changelog.ldif
```

3. Start all instances on the other servers in the replication topology using the following command:

```
# dsctl instance_name start
```

## Verification

To verify that the changelog is encrypted, run the following steps on the server with the encrypted changelog:

1. Make a change in the LDAP directory, such as updating an entry.
2. Stop the instance:

```
# dsctl stop instance_name
```

3. Enter the following command to display parts of the changelog:

```
# dbscan -f /var/lib/dirsrv/slapd-instance_name/changelogdb/replica_name_repl/Gen.db | tail -50
```

If the changelog is encrypted, you see only encrypted data.

4. Start the instance:

```
# dsctl start instance_name
```

## Additional Resources

- [Section 15.15, “Exporting the Replication Changelog”](#)
- [Section 15.16, “Importing the Replication Changelog from an LDIF-formatted Changelog Dump”](#)

## 15.14. REMOVING THE CHANGELOG

The changelog is a record of all modifications on a given replica that the supplier uses to replay these modifications to replicas on consumer servers (or suppliers in the case of multi-supplier replication).

If a supplier server goes offline, it is important to be able to delete the changelog because it no longer holds a true record of all modifications and, as a result, should not be used as a basis for replication. A changelog can be effectively deleted by deleting the log file.

### 15.14.1. Removing the Changelog using the Command Line

To remove the changelog from the supplier server:

1. Verify whether replication is disabled for all suffixes:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication list
There are no replicated suffixes
```

2. Remove the changelog:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication delete-changelog
```

### 15.14.2. Removing the Changelog using the Web Console

To remove the changelog from the supplier server:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Replication** menu, and select the **Replication Changelog** entry.
4. Click **Delete Changelog**.

## 15.15. EXPORTING THE REPLICATION CHANGELOG

In certain situations, such as when you want to encrypt the replication changelog, you must export the changelog as part of the process. Complete this procedure to export the changelog.

### Prerequisites

- Replication is enabled on the Directory Server instance.

### Procedure

To export the changelog to the **/tmp/changelog.ldif** file, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication dump-changelog -o /tmp/changelog.ldif
```

Note that the **dirsrv** user requires appropriate file system permissions to create the specified file.

## 15.16. IMPORTING THE REPLICATION CHANGELOG FROM AN LDIF-FORMATTED CHANGELOG DUMP

Complete this procedure to import an LDIF-formatted replication changelog dump into Directory Server.

### Prerequisites

- Replication is enabled on the Directory Server instance.
- The changelog dump has been created as described in [Section 15.15, “Exporting the Replication Changelog”](#).

### Procedure

To import the changelog dump from the **/tmp/changelog.ldif** file, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication restore-changelog from-ldif /tmp/changelog.ldif
```

Note that the **dirsrv** user requires permissions to read the specified file.

## 15.17. MOVING THE REPLICATION CHANGELOG DIRECTORY

In certain situations, you might want to change the Directory Server replication changelog directory. For example, to change the directory to `/var/lib/dirsrv/slapd-instance_name/new_changelogdb/`:

1. Display the current path to the changelog and set the new path:

- o Using the command line:

1. Display the current directory:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
  -b "cn=changelog5,cn=config" nsslapd-changelogdir
...
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb/
```

You need the displayed path in a later step to move the directory.

2. Set the new path:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogdir
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/new_changelogdb/
```

- o Using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).

2. Select the instance.

3. Open the **Replication** menu, and select the **Replication Changelog** entry.

4. Click **Show Advanced Settings**.

5. Identify the current path in the **Changelog Location** field. You need the displayed path in a later step to move the directory.

6. Set the new path in the **Changelog Location** field.

7. Click **Save**.

2. Stop the Directory Server instance:

```
# dsctl instance_name stop
```

3. Move the content of the previous directory to `/var/lib/dirsrv/slapd-instance_name/new_changelogdb/`:

```
# mv /var/lib/dirsrv/slapd-instance_name/changelogdb/ \
  /var/lib/dirsrv/slapd-instance_name/new_changelogdb/
```

4. Delete the previous directory:

```
# rm /var/lib/dirsrv/slapd-instance_name/changelogdb/
```

5. Start the Directory Server instance:

```
# dsctl instance_name start
```

## 15.18. TRIMMING THE REPLICATION CHANGELOG

The Directory Server changelog manages a list of received and processed changes. It includes changes of clients ran on the server and additionally directory changes from other replication partners. Using the default settings, Directory Server does not automatically remove entries and the changelog grows infinitely. To control which entries are removed, use the following parameters:

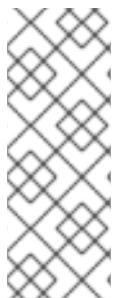
- ***nsslapd-changelogmaxage*** (recommended): Removes entries if they exceed the time set in this parameter.
- ***nsslapd-changelogmaxentries***: Removes the oldest entries if the total number of records exceed the value set in this parameter.

Any record and all subsequently created records remains in the changelog until it is successfully replicated to all servers in the topology. For example, this occurs in a situation when a Directory Server supplier was removed from the topology, but the replica update vector (RUV) had not been removed.

### 15.18.1. Enabling Replication Changelog Trimming

To enable replication changelog trimming and automatically remove entries that are older than 7 days (**7d**):

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 7d
```



#### NOTE

Red Hat recommends setting a maximum age in ***nsslapd-changelogmaxage*** instead of a maximum number of entries in ***nsslapd-changelogmaxentries***. The time set in ***nsslapd-changelogmaxage*** should match the replication purge delay set in ***nsDS5ReplicaPurgeDelay***. For details about ***nsDS5ReplicaPurgeDelay***, see the parameter description in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

### 15.18.2. Manually Reducing the Size of a Large Changelog

When trimming the replication changelog was not enabled and the database grew to a large size, reduce the changelog size manually in the short term:

1. To be able to reset the parameters after reducing the changelog size, display the current values of corresponding parameters. For example:

```
# ldapsearch -x -D "cn=Directory Manager" -W -b "cn=changelog5,cn=config" \
    nsslapd-changelogmaxage nsslapd-changelogcompactdb-interval \
    nsslapd-changelogtrim-interval nsslapd-changelogmaxage

dn: cn=changelog5,cn=config
nsslapd-changelogmaxage: 7d
nsslapd-changelogcompactdb-interval: 2592000
nsslapd-changelogtrim-interval: 300
```

Parameters that are not displayed in the output are not set and Directory Server uses their default values. For the default values, see the parameter descriptions in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

2. Reduce the following parameter's value temporarily:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 3d
-
replace: nsslapd-changelogtrim-interval
nsslapd-changelogtrim-interval: 30
-
replace: nsslapd-changelogcompactdb-interval
nsslapd-changelogcompactdb-interval: 300
```

Using these settings, Directory Server removes changelog entries older than 3 days (***nsslapd-changelogmaxage***) within the next 30 seconds (***nsslapd-changelogtrim-interval***).

3. Restart the Directory Server instance so that the new value of the ***nsslapd-changelogcompactdb-interval*** parameter takes effect:

```
# dsctl instance_name restart
```

After the next database update, the database is automatically compacted within the time interval set in the ***nsslapd-changelogcompactdb-interval*** parameter.

4. Reset the updated parameters to their previous values. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 7d
-
replace: nsslapd-changelogtrim-interval
nsslapd-changelogtrim-interval: 300
-
replace: nsslapd-changelogcompactdb-interval
nsslapd-changelogcompactdb-interval: 2592000
```

**IMPORTANT**

For performance reasons, do not permanently use too short interval settings.

5. Restart the Directory Server instance:

```
# dsctl instance_name restart
```

## 15.19. FORCING REPLICATION UPDATES

If you stop a Directory Server instance that is involved in replication for regular maintenance, you must update the replication immediately when it comes back online. In the case of a supplier in a multi-supplier environment, the directory information needs to be updated by an other supplier in the setup. In other cases, for example if you take down a hub or a dedicated consumer for maintenance and then come back online later, the supplier server needs to update the status.

### Prerequisites

- The replication is set up.
- The consumer has been initialized.

### Procedure

1. Check if the replication agreement has an update schedule configured:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt get --suffix="dc=example,dc=com" agreement_name
```

If the output of the command contains **nsDS5ReplicaUpdateSchedule: \*** or the **nsDS5ReplicaUpdateSchedule** parameter is not present, no update schedule is configured.

If **nsDS5ReplicaUpdateSchedule** contains a schedule, such as shown below, note the value:

```
nsDS5ReplicaUpdateSchedule: 0800-2200 0246
```

2. If an update schedule is configured, enter the following command to temporary disable it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt set --schedule \'* --suffix="dc=example,dc=com" agreement_name
```

3. Temporarily disable the replication agreement:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt disable --suffix="dc=example,dc=com" agreement_name
```

4. Re-enable the replication agreement to force the replication update:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt enable --suffix="dc=example,dc=com" agreement_name
```

5. If a replication schedule was configured at the beginning of this procedure, set the schedule to the previous value:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt set --schedule
"0800-2200 0246" --suffix="dc=example,dc=com" agreement_name
```

## 15.20. SETTING REPLICATION TIMEOUT PERIODS

Suppliers must have an exclusive connection to a consumer to send updates to the directory. As mentioned in [Section 15.4.3, “Preventing Monopolization of a Consumer in Multi-Supplier Replication”](#), it is possible to configure a wait time for suppliers attempting to connect to a consumer, so that the supplier does not hang while the consumer is tied up with another supplier.

It is also possible to set a timeout period for a supplier, so that it does not stay connected to a consumer interminably attempting to send updates over a slow or broken connection.

There are two attributes which set the timeout period:

- ***nsDS5ReplicaTimeout*** sets the number of seconds that the replication operation waits for a response from the consumer before timing out and failing. To set the optimum number, check the access logs to see the average amount of time that the replication process takes, and set the timeout period accordingly.
- ***nsDS5DebugReplicaTimeout*** sets the timeout period for the replication operation when debug logging is enabled. This setting may be appreciably higher than the ***nsDS5ReplicaTimeout*** setting because debug logging can slow down directory operations. This attribute can optionally set an error log level where this parameter is applied; the default is replication debugging (8192).

Both of these attributes are set in the replication agreement of the replicated suffix:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=example-agreement,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
add: nsDS5ReplicaTimeout
nsDS5ReplicaTimeout: 600
-
add: nsDS5DebugReplicaTimeout
nsDS5DebugReplicaTimeout: 6000
```

## 15.21. USING THE RETRO CHANGELOG PLUG-IN

The Retro Changelog plug-in configures Directory Server to maintain a changelog that is compatible with the changelog implemented in Directory Server 4.x.



### NOTE

Only enable the Retro Changelog plug-in if you need to maintain a changelog for directory clients that depend on a Directory Server 4.x-style changelog.

To use the retro changelog plug-in, the Directory Server instance must be configured as a single-supplier replica.

When the Directory Server is configured to maintain a retro changelog, this changelog is stored in a separate database under a special suffix, **cn=changelog**.

The retro changelog consists of a single level of entries. Each entry in the changelog has the object class **changeLogEntry**. For a list of possible attributes in a changelog entry, see the [Changelog Attributes](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

### 15.21.1. Enabling the Retro Changelog Plug-in

This section describes how to enable the **Retro Changelog** plug-in:

#### 15.21.1.1. Enabling the Retro Changelog Plug-in Using the Command Line

To enable the **Retro Changelog** plug-in using the command line:

1. Use the **dsconf** utility to enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin retro-changelog
enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

#### 15.21.1.2. Enabling the Retro Changelog Plug-in Using the Web Console

To enable the **Retro Changelog** plug-in using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Select the **Plugins** menu.
4. Select the **Retro Changelog** plug-in in the list on the left.
5. Change the status to **On**.
6. Click **Save Config**.
7. Restart the instance. See [Section 1.5.2, “Starting and Stopping a Directory Server Instance Using the Web Console”](#).

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 15.21.2. Trimming the Retro Changelog

The size of the retro changelog is automatically reduced if you lower the maximum age of records set in the **nsslapd-changelogmaxage** parameter and the next trim interval, set in **nsslapd-changelog-trim-interval**, is executed.

For example, to set maximum age of records in the retro changelog to two days:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin retro-changelog set --max-age="2d"
```

### 15.21.3. Searching and Modifying the Retro Changelog

The changelog supports search operations and is optimized for searches that include filters of the form **(&(changeNumber>=X)(changeNumber<=Y))**.

As a general rule, do not perform add or modify operations on the retro changelog entries, although entries can be deleted to trim the size of the changelog. Only modify the retro changelog entry to modify the default access control policy.

### 15.21.4. Retro Changelog and the Access Control Policy

When the retro changelog is created, the following access control rules apply by default:

- Read, search, and compare rights are granted to all authenticated users (**userdn=anyone**, not to be confused with anonymous access where **userdn=all**) to the retro changelog top entry **cn=changelog**.
- Write and delete access are not granted, except implicitly to the Directory Manager.

Do not grant read access to anonymous users because the changelog entries can contain modifications to sensitive information, such as passwords. Only authenticated applications and users should be allowed to access this information.

To modify the default access control policy which applies to the retro changelog, modify the **aci** attribute of the **cn=changelog** entry.

## 15.22. DISPLAYING THE STATUS OF A SPECIFIC REPLICATION AGREEMENT

You can display the status of a specific replication agreement using the command line and in the web console.

### 15.22.1. Displaying the Status of a Specific Replication Agreement Using the Command-Line

To display the status of a replication agreement using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt status --suffix="dc=example,dc=com" agreement_name
agreement_name Status for agreement_name agmt consumer.example.com:636
Replica Enabled: on
Update In Progress: FALSE
Last Update Start: 19700101000000Z
Last Update End: 19700101000000Z
Number Of Changes Sent: 0
Number Of Changes Skipped: None
Last Update Status: Error (-1) Problem connecting to replica - LDAP error: Can't contact LDAP server (connection error)
```

Init In Progress:  
 Last Init Start: 19700101000000Z  
 Last Init End: 19700101000000Z  
 Last Init Status: unavailable  
 Reap Active: 0  
 Replication Status: Not in Synchronization: supplier (5bed8467000100010000) consumer (Unavailable) Reason(Unknown)  
 Replication Lag Time: Unavailable

This example shows replication currently fails because the **consumer.example.com** host is unavailable.

### 15.22.2. Displaying the Status of a Specific Replication Agreement Using the Web Console

To display the status of a replication agreement in the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Monitoring** menu, and select **Replication**
4. Select **Replication** entry from the list in the left pane.
5. Depending on whether you want to display the status of a replication agreement between Directory Server instances or a Winsync Agreement, select the appropriate tab.

Agreement	Replica	Enabled	Update Status	Changes Sent
example-agreement	consumer.example.com:389	on	Error (0) Replica acquired successfully: Incremental update succeeded	1:151/0

The **Update Status** column displays the status of replication agreement.

## 15.23. MONITORING THE REPLICATION TOPOLOGY

Use the **dsconf replication monitor** command to display the replication status, as well as additional information, such as replica IDs and Change State Numbers (CSN) on suppliers, consumers, and hubs:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication monitor
Enter password for cn=Directory Manager on ldap://supplier.example.com: password
```

```
Enter a bind DN for consumer.example.com:389: cn=Directory Manager
Enter a password for cn=Directory Manager on consumer.example.com:389: password
Supplier: server.example.com:389
```

```
-----
Replica Root: dc=example,dc=com
Replica ID: 1
Replica Status: Available
Max CSN: 5e3acb77001d00010000
```

```

Status For Agreement: "example-agreement" (consumer.example.com:389)
Replica Enabled: on
Update In Progress: FALSE
Last Update Start: 20200205140439Z
Last Update End: 20200205140440Z
Number Of Changes Sent: 1:166/0
Number Of Changes Skipped: None
Last Update Status: Error (0) Replica acquired successfully: Incremental update succeeded
Last Init Start: 20200205133709Z
Last Init End: 20200205133711Z
Last Init Status: Error (0) Total update succeeded
Reap Active: 0
Replication Status: In Synchronization
Replication Lag Time: 00:00:00

Supplier: consumer.example.com:389
-----
Replica Root: dc=example,dc=com
Replica ID: 65535
Replica Status: Available
Max CSN: 00000000000000000000000000000000

```

### 15.23.1. Setting Credentials for Replication Monitoring in the .dsrc File

By default, the **dsconf replication monitor** command prompts for bind DNs and passwords when authenticating to remote instances. Alternatively, you can set the bind DNs, and optionally passwords, for each server in the topology in the user's `~/.dsrc` file.

#### Example 15.1. An Example .dsrc File with Explanations of the Different Fields

The following is an example `~/.dsrc` file:

```

[repl-monitor-connections]
connection1 = server1.example.com:389:cn=Directory Manager:*
connection2 = server2.example.com:389:cn=Directory Manager:[~/pwd.txt]
connection3 = hub1.example.com:389:cn=Directory Manager:S3cret

```

This example uses **connection1** to **connection3** as keys for each entry. However, you can use any key as long as it is unique.

If you run the **dsconf replication monitor** command, the **dsconf** utility connects to all servers configured in replication agreements of the instance. If the utility finds the host name in `~/.dsrc`, it uses the defined credentials to authenticate to the remote server. In the example above, **dsconf** uses the following credentials when connecting to a server:

Host name	Bind DN	Password
<b>server1.example.com</b>	<b>cn=Directory Manager</b>	Prompts for the password
<b>server2.example.com</b>	<b>cn=Directory Manager</b>	Reads password from <code>~/pwd.txt</code>
<b>hub1.example.com</b>	<b>cn=Directory Manager</b>	<b>S3cret</b>

### 15.23.2. Using Aliases in the Replication Topology Monitoring Output

By default, the **dsconf replication monitor** command displays the host names of servers in the monitoring report. Alternatively, you can display aliases using one of the following methods:

- Define the aliases in the `~/.dsrc` file:

```
[repl-monitor-aliases]
M1 = server1.example.com:389
M2 = server2.example.com:389
```

- Define the aliases by passing the `-a alias=host_name:port` parameter to the **dsconf replication monitor** command:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com replication monitor -a
M1=server1.example.com:389 M2=server2.example.com:389
```

In both cases, the command displays the alias in the command's output:

```
...
Supplier: M1 (server1.example.com:389)
...
Supplier: M2 (server2.example.com:389)
...
```

### 15.24. COMPARING TWO DIRECTORY SERVER INSTANCES

In certain situations, an administrator wants to compare if two Directory Servers are synchronized. The **ds-replcheck** utility enables you to compare two online servers. Alternatively, **ds-replcheck** can compare two LDIF-formatted files in offline mode and two servers in online mode.



#### NOTE

To compare two databases offline, export them using the **db2ldif -r** command to include replication state information.

If you compare two online servers, the contents of the databases usually differ, if they are under heavy load. To work around this problem, the script uses a lag time value in by passing the `-l time_in_seconds` parameter to **ds-replcheck**. By default, this value is set to **300** seconds (5 minutes). If the utility detects an inconsistency that is within the lag time, it is not reported. This helps to reduce false positives.

By default, if you excluded certain attributes in the replication agreement from being replicated, **ds-replcheck** reports these attributes as different. To ignore these attributes, pass the `-i attribute_list` parameter to the utility.

For example, to compare the **dc=example,dc=com** suffix of two Directory Servers:

```
# ds-replcheck -D "cn=Directory Manager" -W \
-m ldap://server1.example.com:389 \
-r ldap://server2.example.com:389 \
```

```
-b "dc=example,dc=com"
```

The output of the utility contains the following sections:

### Database RUV's

Lists the Replication Update Vectors (RUV) of the databases including the minimum and maximum Change Sequence Numbers (CSN). For example:

Supplier RUV:

```
{replica 1 ldap://server1.example.com:389} 58e53b92000200010000 58e6ab46000000010000
{replica 2 ldap://server2.example.com:389} 58e53baa000000020000 58e69d7e000000020000
{replicageneration} 58e53b7a000000010000
```

Replica RUV:

```
{replica 1 ldap://server1.example.com:389} 58e53ba1000000010000 58e6ab46000000010000
{replica 2 ldap://server2.example.com:389} 58e53baa000000020000 58e7e8a3000000020000
{replicageneration} 58e53b7a000000010000
```

### Entry Count

Displays the total number of entries on the both servers, including tombstone entries. For example:

Supplier: 12

Replica: 10

### Tombstones

Displays the number of tombstone entries on each replica. These entries are added to the total entry count. For example:

Supplier: 4

Replica: 2

### Conflict Entries

Lists the Distinguished Names (DN) of each conflict entry, the conflict type, and the date it was created. For example:

Supplier Conflict Entries: 1

- nsuniqueid=48177227-2ab611e7-afcb801a-ecef6d49+uid=user1,dc=example,dc=com
  - Conflict: namingConflict (add) uid=user1,dc=example,dc=com
  - Glue entry: no
  - Created: Wed Apr 26 20:27:40 2017

Replica Conflict Entries: 1

- nsuniqueid=48177227-2ab611e7-afcb801a-ecef6d49+uid=user1,dc=example,dc=com
  - Conflict: namingConflict (add) uid=user1,dc=example,dc=com
  - Glue entry: no
  - Created: Wed Apr 26 20:27:40 2017

### Missing Entries

Lists the DNs of each missing entry and the creation date from the other server where the entry resides. For example:

Entries missing on Supplier:

- uid=user2,dc=example,dc=com (Created on Replica at: Wed Apr 12 14:43:24 2017)
- uid=user3,dc=example,dc=com (Created on Replica at: Wed Apr 12 14:43:24 2017)

Entries missing on Replica:

- uid=user4,dc=example,dc=com (Created on Supplier at: Wed Apr 12 14:43:24 2017)

## Entry Inconsistencies

Lists the DNs of the entry that contain attributes that are different to those on the other server. If a state information is available, it is also displayed. If no state information for an attribute is available, it is listed as an origin value. This means that the value was not updated since the replication was initialized for the first time. For example:

```
cn=group1,dc=example,dc=com
-----
```

Replica missing attribute "objectclass":

- Supplier's State Info: objectClass;vucsn-58e53baa000000020000: top
- Date: Wed Apr 5 14:47:06 2017
  
- Supplier's State Info: objectClass;vucsn-58e53baa000000020000: groupofuniqueNames
- Date: Wed Apr 5 14:47:06 2017

## 15.25. SOLVING COMMON REPLICATION CONFLICTS

Multi-supplier replication uses an eventually-consistency replication model. This means that the same entries can be changed on different servers. When replication occurs between these two servers, the conflicting changes need to be resolved. Mostly, resolution occurs automatically, based on the time stamp associated with the change on each server. The most recent change takes precedence.

However, there are some cases where conflicts require manual intervention in order to reach a resolution. Entries with a change conflict that cannot be resolved automatically by the replication process.

To list conflict entries, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict list dc=example,dc=com
```

### 15.25.1. Solving Naming Conflicts

When two entries are created with the same DN on different servers, the automatic conflict resolution procedure during replication renames the last entry created, including the entry's unique identifier in the DN. Every directory entry includes a unique identifier stored in the **nsuniqueid** operational attribute. When a naming conflict occurs, this unique ID is appended to the non-unique DN.

For example, if the **uid=user\_name,ou=People,dc=example,dc=com** entry was created on two different servers, replication adds the unique ID to the DN of the last entry created. This means, the following entries exist:

- **uid=user\_name,ou=People,dc=example,dc=com**
- **nsuniqueid=66446001-1dd211b2+uid=user\_name,ou=People,dc=example,dc=com**

To resolve the replication conflict, you must manually decide how to proceed:

- To keep only the valid entry (**uid=user\_name,ou=People,dc=example,dc=com**) and delete the conflict entry, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict delete
nsuniqueid=66446001-1dd211b2+uid=user_name,ou=People,dc=example,dc=com
```

- To keep only the conflict entry (**nsuniqueid=66446001-1dd211b2+uid=user\_name,ou=People,dc=example,dc=com**), enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict swap
nsuniqueid=66446001-1dd211b2+uid=user_name,ou=People,dc=example,dc=com
```

- To keep both entries, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict convert --new-
rdn=uid=user_name_NEW nsuniqueid=66446001-
1dd211b2+uid=user_name,ou=People,dc=example,dc=com
```

To keep the conflict entry, you must specify a new Relative Distinguished Name (RDN) in the **--new-rdn** option to rename the conflict entry. The previous command renames the conflict entry to **uid=user\_name\_NEW,ou=People,dc=example,dc=com**.

### 15.25.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a **glue** entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

**Glue entries** are temporary entries that include the object classes **glue** and **extensibleObject**. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the **glue** object class and the **nsds5RepConflict** attribute.

In such cases, either modify the glue entry to remove the **glue** object class and the **nsds5RepConflict** attribute to keep the entry as a normal entry or delete the glue entry and its child entries.

- The server creates a minimalistic entry with the **glue** and **extensibleObject** object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

To list all glue entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict list-glue suffix
```

To delete a glue entry and its child entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict delete-glue DN_of_glue_entry
```

To convert a glue entry into a regular entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-conflict convert-glue DN_of_glue_entry
```

### 15.25.3. Resolving Errors for Obsolete or Missing Suppliers

Information about the replication topology, that is all suppliers which supply updates to each other and other replicas within the same replication group, is contained in a set of metadata called the *replica update vector (RUV)*. The RUV contains information about the supplier such as its ID and URL, its latest change state number (CSN) on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

When one supplier is removed from the replication topology, it may remain in another replica's RUV. When the other replica is restarted, it can record errors in its log, warning that the replication plug-in does not recognize the removed supplier. The errors will look similar to the following example:

```
[22/Jan/2021:17:16:01 -0500] NSMMReplicationPlugin - ruv_compare_ruv: RUV [changelog max RUV] does not contain element [{replica 8 ldap://m2.example.com:389} 4aac3e59000000080000 4c6f2a02000000080000] which is present in RUV [database RUV]
```

<...several more samples...>

```
[22/Jan/2021:17:16:01 -0500] NSMMReplicationPlugin - replica_check_for_data_reload: Warning: for replica dc=example,dc=com there were some differences between the changelog max RUV and the database RUV. If there are obsolete elements in the database RUV, you should remove them using the CLEANALLRUV task. If they are not obsolete, you should check their status to see why there are no changes from those servers in the changelog.
```

Note which replica and its ID; in this case, replica **8**.

When the supplier is permanently removed from the topology, then any lingering metadata about that supplier should be purged from every other supplier's RUV entry. Use the **cleanallruv** directory task to remove a RUV entry from all suppliers in the topology.



#### NOTE

The **cleanallruv** task is replicated. Therefore, you only need to run it on one supplier.

#### Procedure 15.1. Removing an Obsolete or Missing Supplier Using the**cleanallruv** Task Operation

1. List all RUV records and replica IDs, both valid and invalid, as deleted suppliers may have left metadata on other suppliers:

```
# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com -D "cn=Directory Manager" -W -b dc=example,dc=com '(&(nsuniqueid=ffffffff-ffff-ffff-ffff-ffff)(objectclass=nstombstone))'
```

```

nsDS5Replicaid nsDS5ReplicaType nsds50ruv
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5Replicaid: 1
nsDS5ReplicaType: 3
nsds50ruv: {replicageneration} 55d5093a000000010000
nsds50ruv: {replica 1 ldap://m1.example.com:389} 55d57026000000010000
55d57275000000010000
nsds50ruv: {replica 20 ldap://m2.example.com:389} 55e74b8c000000140000
55e74bf7000000140000
nsds50ruv: {replica 9 ldap://m2.example.com:389}
nsds50ruv: {replica 8 ldap://m2.example.com:389} 506f921f000000080000
50774211000500080000

```

Note the returned replica IDs: **1**, **20**, **9**, and **8**.

2. List the currently defined and valid replica IDs of all suppliers which are replicating databases by searching the replica configuration entries DN **cn=replica** under the **cn=config** suffix.



#### NOTE

Consumers and read-only nodes always have the replica ID set to **65535**, and **nsDS5ReplicaType: 3** signifies a supplier.

```

# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com m2.example.com -D "cn=Directory
Manager" -W -b cn=config cn=replica nsDS5Replicaid nsDS5ReplicaType
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5Replicaid: 1
nsDS5ReplicaType: 3

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5Replicaid: 20
nsDS5ReplicaType: 3

```

After you search all URIs returned in the first step (in this procedure, **m1.example.com** and **m2.example.com**), compare the list of returned suppliers (entries which have **nsDS5ReplicaType: 3**) to the list of RUVs from the previous step. In the above example, this search only returned IDs **1** and **20**, but the previous search also returned **9** and **8** on URI **m2.example.com**. This means that the latter two are removed, and their RUVs need to be cleaned.

3. After determining which RUVs require cleaning, create a new **cn=cleanallruv,cn=tasks,cn=config** entry and provide the following information about your replication configuration:
  - The base DN of the replicated database (**replica-base-dn**)
  - The replica ID (**replica-id**)
  - Whether to catch up to the maximum change state number (CSN) from the missing supplier, or whether to just remove all RUV entries and miss any updates (**replica-force-cleaning**); setting this attribute to **no** means that the task will wait for all the configured replicas to catch up with all the changes from the removed replica first, and then remove the RUV.

```
# dsconf -D "cn=Directory Manager" ldap://m2.example.com repl-tasks \
cleanallruv --suffix="dc=example,dc=com" --replica-id=8
```

**NOTE**

The **cleanallruv** task is replicated. Therefore, you only need to run it on one supplier.

Repeat the same for every RUV you want to clean (ID **9** in this procedure).

4. After cleaning the RUVs of all replicas discovered earlier, you can again use the search from the first step to verify that all extra RUVs are removed:

```
# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com -D "cn=Directory Manager" -W -b
dc=example,dc=com '(&(nsuniqueid=ffffffff-ffff-ffff-ffff-ffffffff)(objectclass=nstombstone))'
nsDS5Replicaid nsDS5ReplicaType nsds50ruv
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5Replicaid: 1
nsDS5ReplicaType: 3
nsds50ruv: {replicageneration} 55d5093a000000010000
nsds50ruv: {replica 1 ldap://m1.example.com:389} 55d57026000000010000
55d57275000000010000
nsds50ruv: {replica 20 ldap://m2.example.com:389} 55e74b8c000000140000
55e74bf7000000140000
```

As you can see in the above output, replica IDs **8** and **9** are no longer present, which indicates that their RUVs have been cleaned successfully.

## 15.26. TROUBLESHOOTING REPLICATION-RELATED PROBLEMS

This section lists some error messages, explains possible causes, and offers remedies.

It is possible to get more debugging information for replication by setting the error log level to **8192**, which is replication debugging. See [Section 21.3.7, “Configuring the Log Levels”](#).

To change the error log level to **8192**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace 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.

### 15.26.1. Possible Replication-related Error Messages

The following sections describe many common replication problems.

#### **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-supplier 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 web console or M1 or M2's error log). Also, M2 should not initialize M1 back.

**Warning: data for replica's was reloaded, and it no 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-supplier replication, reinitialize the consumers. Otherwise, if the server later complains that it cannot locate some CSN for a consumer, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

**agmt=%s(%s:%d): 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-supplier replication, reinitialize the consumers. Otherwise, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

## 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.

**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.

### 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:

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

dn: cn=changelog5,cn=config
changetype: modify
add: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 1d
```

**1d** means 1 day. Other valid time units are s for seconds, m for minutes, h for hours, and w for weeks. A value of 0 turns off the purge.

With changelog purge turned on, a purge thread that wakes up every five minutes will remove a change if its age is greater than the value of nsslapd-changelogmaxage and if it has been replayed to all the direct consumers of this supplier (supplier or hub).

If it appears that the changelog is not purged when the purge threshold is reached, check the maximum time lag from the replication monitor among all the consumers. Irrespective of what the purge threshold is, no change will be purged before it is replayed by all the consumers.

### The Replication Monitor is not responding.

- Reason: The LDAPS 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 LDAPS port problem, one of the servers in the replication topology might hang.
- Remedy: Map the TLS port to a non-TLS port in the configuration file of the Replication Monitor. For example, if 636 is the TLS port and 389 is the non-TLS port, add the following line in the **[connection]** section:

```
*:636=389:*:password
```

### In the Replication Monitor, some consumers show just the header of the table.

- 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.

# CHAPTER 16. SYNCHRONIZING RED HAT DIRECTORY SERVER WITH MICROSOFT ACTIVE DIRECTORY

Windows Synchronization carries over changes in a directory – adds, deletes, and changes in groups, users, and passwords – between Red Hat Directory Server and Microsoft Active Directory. This makes it much more efficient and effective to maintain consistent information across directories.

## 16.1. ABOUT WINDOWS SYNCHRONIZATION

Synchronization allows the user and group entries in Active Directory to be matched with the entries in the Red Hat Directory Server. As entries are created, modified, or deleted, the corresponding change is made to the sync peer server, allowing two-way synchronization of users, passwords, and groups.

The synchronization process is analogous to the replication process: the synchronization is enabled by a plug-in, configured and initiated through a sync agreement, and record of directory changes is maintained and updates are sent according to that changelog. This synchronizes users and groups between Directory Server and a Windows server.

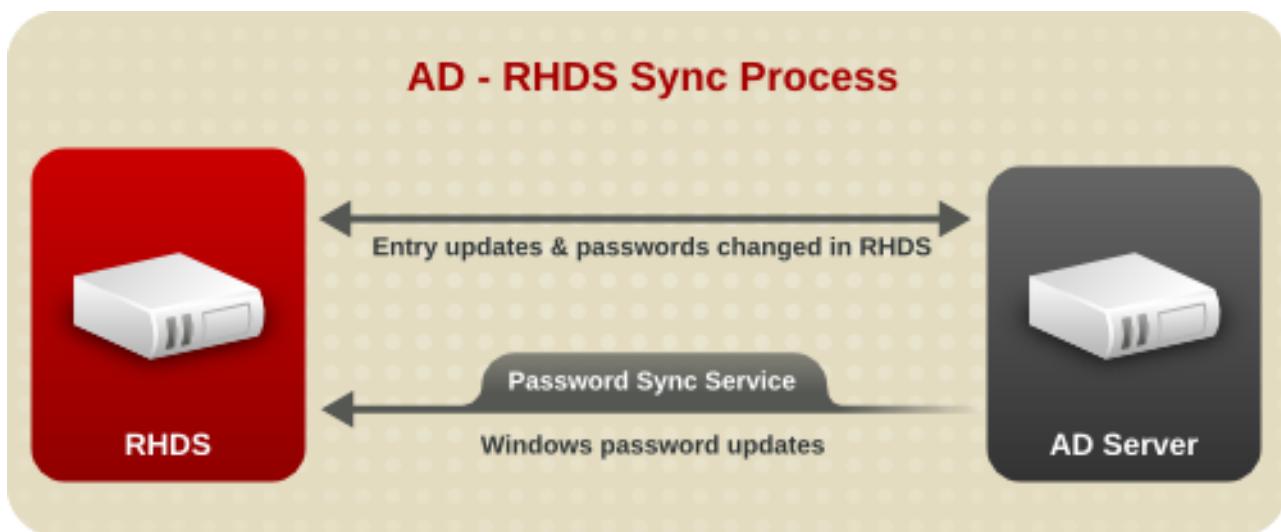
Windows Synchronization has two parts, one for user and group entries and the other for passwords:

- *Directory Server Windows Synchronization.* Synchronization for user and group entries is configured in a synchronization agreement, much like replication is configured in a replication agreement. A sync agreement defines what kinds of entries are synchronized (users, groups, or both) and which direction changes are synchronized (from the Directory Server to Active Directory, from Active Directory to Directory Server, or both).

The Directory Server relies on the Multi-Supplier Replication Plug-in to synchronize user and group entries. The same changelog that is used for multi-supplier replication is also used to send updates from the Directory Server to Active Directory as LDAP operations. The server also performs LDAP search operations against its Windows server to synchronize changes made to Windows entries to the corresponding Directory Server entry.

- *Password Synchronization Service.* If you set the ***nsslapd-unhashed-pw-switch*** parameter in the ***cn=config*** entry to ***on***, password changes made on Directory Server are automatically synchronized over to Active Directory. However, there must be a special hook to recognize and transmit password changes on Active Directory over to Directory Server. This is done by the Password Synchronization Service. This application captures password changes on the Active Directory domain controller and sends them to the Directory Server over LDAPS.

**The Password Synchronization Service must be installed on every Active Directory domain controller.**



**Figure 16.1. Active Directory – Directory Server Synchronization Process**

Synchronization is configured and controlled by one or more *synchronization agreements*, which establishes synchronization between *sync peers*, the directory servers being synchronized. These are similar in purpose to replication agreements and contain a similar set of information, including the host name (or IPv4 or IPv6 address) and port number for Active Directory. The Directory Server connects to its peer Windows server using LDAP/LDAPS to both send and receive updates.

LDAP, a standard connection, can be used for syncing user and group entries alone, but to synchronize passwords, some sort of secure connection is required. If a secure connection is not used, the Windows domain will not accept password changes from the Directory Server and the Password Synchronization Service will not send passwords from the Active Directory domain to the Directory Server. Windows Synchronization allows both LDAPS using TLS and STARTTLS.

Multiple subtree pairs can be configured to sync each other. Unlike replication, which connects *databases*, synchronization is between *suffixes*, parts of the directory tree structure. The synchronized Active Directory and Directory Server suffixes are both specified in the sync agreement. All entries within the respective subtrees are candidates for synchronization, including entries that are not immediate children of the specified suffix DN.



#### NOTE

Any descendant container entries need to be created separately in Active Directory by an administrator; Windows Synchronization does not create container entries.

The Directory Server maintains a *changelog*, a database that records modifications that have occurred. The changelog is used by Windows Synchronization to coordinate and send changes made to the Active Directory peer. Changes to entries in Active Directory are found by using Active Directory's Dirsync search feature. Directory Server runs the Dirsync search periodically by default every five minutes to check for changes on the Active Directory server. You can change this default by setting the ***winSyncInterval*** parameter in the ***cn=syncAgreement\_Name,cn=WindowsReplica,cn=suffix\_Name,cn=mapping tree,cn=config*** entry. Using Dirsync ensures that only those entries that have changed since the previous search are retrieved.

In some situations, such as when synchronization is configured or there have been major changes to directory data, a total update, or *resynchronization*, can be run. This examines every entry in both sync peers and sends any modifications or missing entries. A full Dirsync search is initiated whenever a total update is run. See [Section 16.11, "Sending Synchronization Updates"](#) for more information.

Windows Synchronization provides some control over which entries are synchronized to grant administrators fine-grained control of the entries that are synchronized and to give sufficient flexibility to support different deployment scenarios. This control is set through different configuration attributes set in the Directory Server:

- When creating the sync agreement, there is an option to synchronizing new Windows entries (**nsDS7NewWinUserSyncEnabled** and **nsDS7NewWinGroupSyncEnabled**) as they are created. If these attributes are set to **on**, then existing Windows users/groups are synchronized to the Directory Server, and users/groups as they are created are synchronized to the Directory Server.

Within the Windows subtree, only entries with user or group object classes can be synchronized to Directory Server.

- On the Directory Server, only entries with the **ntUser** or **ntGroup** object classes and attributes can be synchronized.

The placement of the sync agreement depends on what suffixes are synchronized; for a single suffix, the sync agreement is made for that suffix alone; for multiple suffixes, the sync agreement is made at a higher branch of the directory tree. To propagate Windows entries and updates throughout the Directory Server deployment, make the agreement between a supplier in a multi-supplier replication environment, and use that supplier to replicate the changes across the Directory Server deployment, as shown in [Figure 16.2, “Multi-Supplier Directory Server – Windows Domain Synchronization”](#).



### IMPORTANT

While it is possible to configure a sync agreement on a hub server, this only allows uni-directional synchronization, from Red Hat Directory Server to Active Directory. The Active Directory server cannot sync any changes back to the hub.

It is strongly recommended that only suppliers in multi-supplier replication be used to configure synchronization agreements.



### WARNING

There can only be a single sync agreement between the Directory Server environment and the Active Directory environment. Multiple sync agreements to the same Active Directory domain can create entry conflicts.

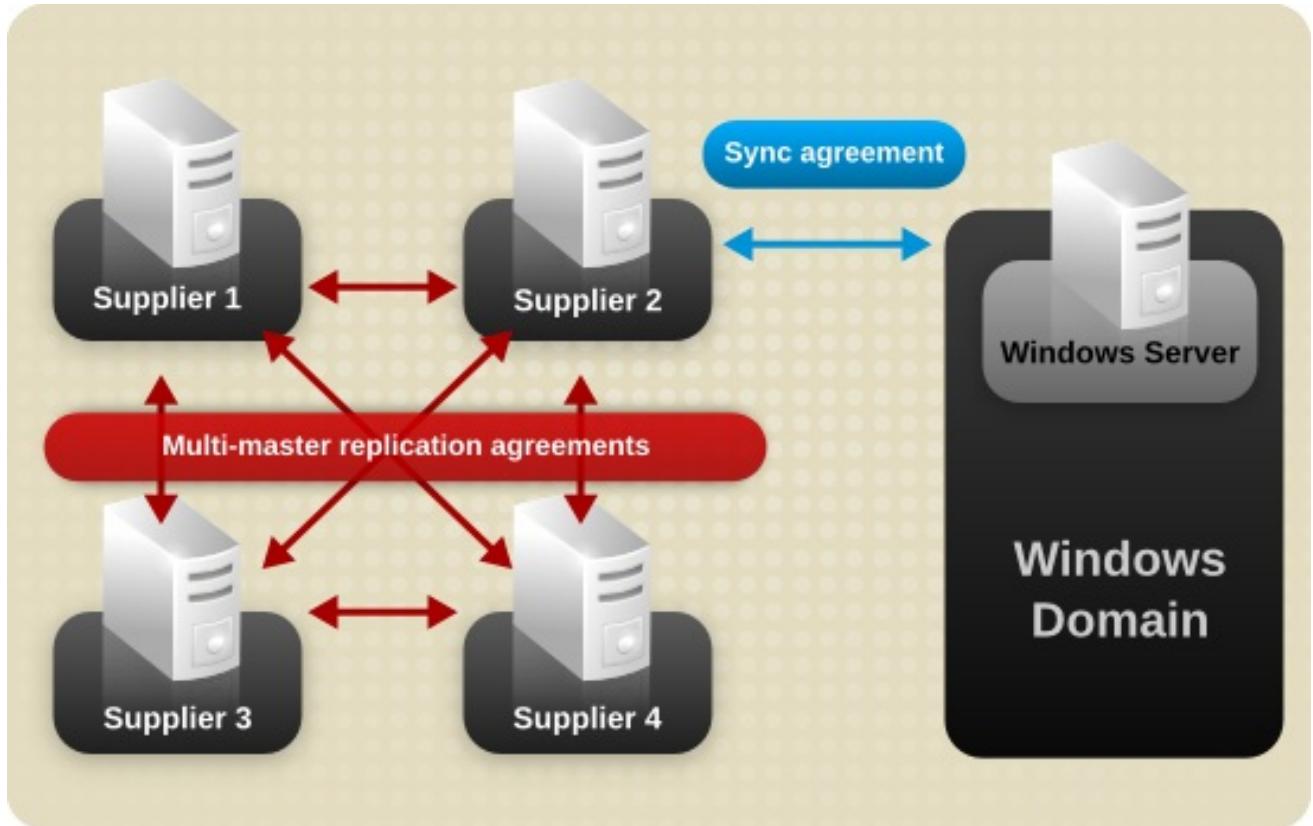


Figure 16.2. Multi-Supplier Directory Server – Windows Domain Synchronization

Directory Server passwords are synchronized along with other entry attributes because plain-text passwords are retained in the Directory Server changelog. The Password Synchronization service is needed to catch password changes made on Active Directory. Without the Password Synchronization service, it would be impossible to have Windows passwords synchronized because passwords are hashed in Active Directory, and the Windows hashing function is incompatible with the one used by Directory Server.

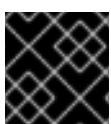
## 16.2. SUPPORTED ACTIVE DIRECTORY VERSIONS

See the corresponding section for the latest Directory Server version in the [Red Hat Directory Server Release Notes](#).

## 16.3. SYNCHRONIZING PASSWORDS

Password changes in a Directory Server entry can be synchronized to password attributes in Active Directory entries by using the **Password Sync** utility.

When passwords are synchronized, password policies are enforced on each sync peer locally. 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 synchronized over to the Windows server, the Windows password policy is enforced.



### IMPORTANT

The password policies themselves are not synchronized.

Configuration information is kept locally and cannot be synchronized, including the password change history and the account lockout counters.

When configuring a password policy for synchronization, 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 instances, configure the Directory Server for multi-supplier replication.
- Password expiration warnings and times, failed bind attempts, and other password-related information is enforced locally per server and is not synchronized between sync peer servers.
- On the Directory Server instance that has the replication agreement with the Windows server configured, set the **nsslapd-unhashed-pw-switch** parameter in the **cn=config** entry to **on**.
- The same bind behavior should occur on all servers. Make sure to create the same or similar password policies on both Directory 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).

## 16.4. SETTING UP SYNCHRONIZATION BETWEEN ACTIVE DIRECTORY AND DIRECTORY SERVER

Configuring synchronization is very similar to configuring replication. It requires configuring the database as a supplier with a changelog and creating an agreement to define synchronization. A common user identity, a synchronization user, connects to the Active Directory (AD) domain controller (DC) to send updates from Directory Server to AD and to check AD for updates to synchronize them to Directory Server.



### NOTE

To enable users to use their accounts on Directory Server and AD, synchronize passwords. Password synchronization requires to use an encrypted connection.

Synchronization for user and group entries is passive from the AD side. Directory Server send updates to AD and polls for updates on the AD domain. For passwords, the AD server requires a separate password service. This service actively sends password changes from the AD domain to Directory Server.

### 16.4.1. Step 1: Enabling TLS on the Directory Server Host

The **Password Sync** service requires to synchronize passwords over an encrypted connection. If TLS is not yet enabled in your Directory Server instance, enable it. For details, see [Section 9.4.1, "Enabling TLS in Directory Server"](#).

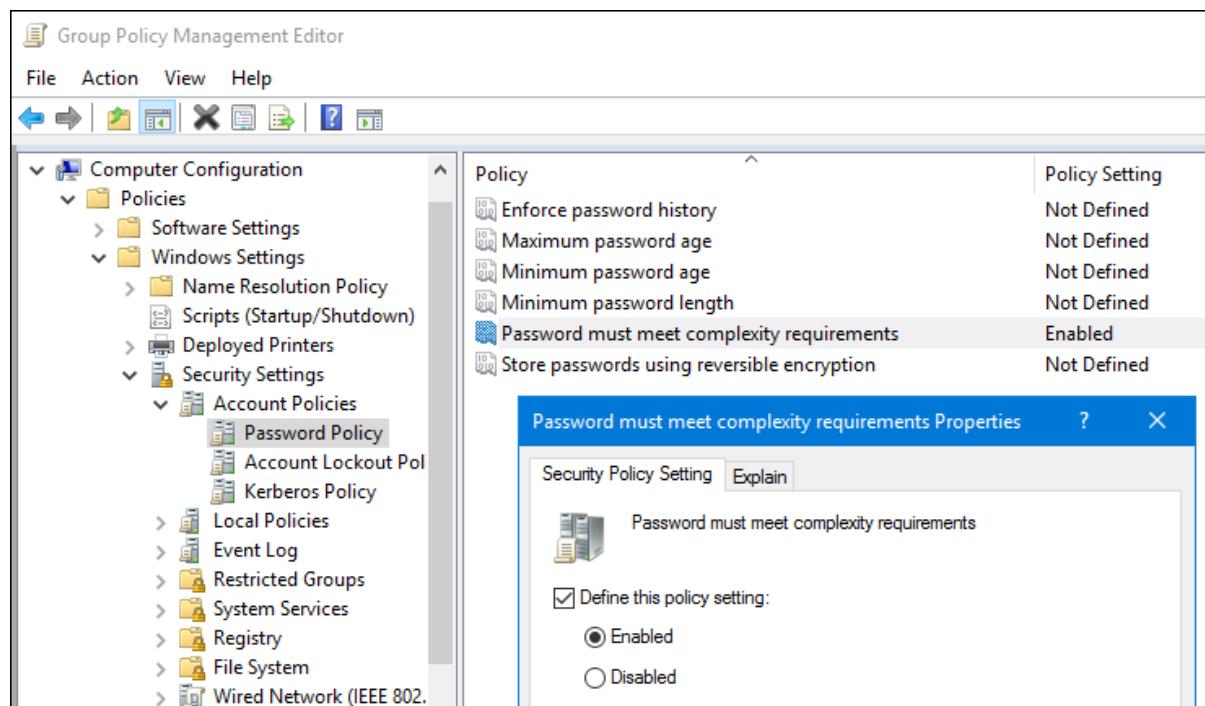
### 16.4.2. Step 2: Enabling Password Complexity in the AD Domain

Enable password complexity in the AD domain using a group policy. For example:

1. Open the **Group Policy Management** console and create a new Group Policy Object (GPO) in the domain.

For details about using the **Group Policy Management** console, see the Windows documentation.

2. Right-click the GPO, and select **Edit** to open the **Group Policy Management Editor**.
3. Navigate to **Computer Configuration → Windows Settings → Security Settings → Account Policies → Password Policy**, and double-click the policy named **Password must meet complexity requirements**.
4. Enable the policy and click **OK**.

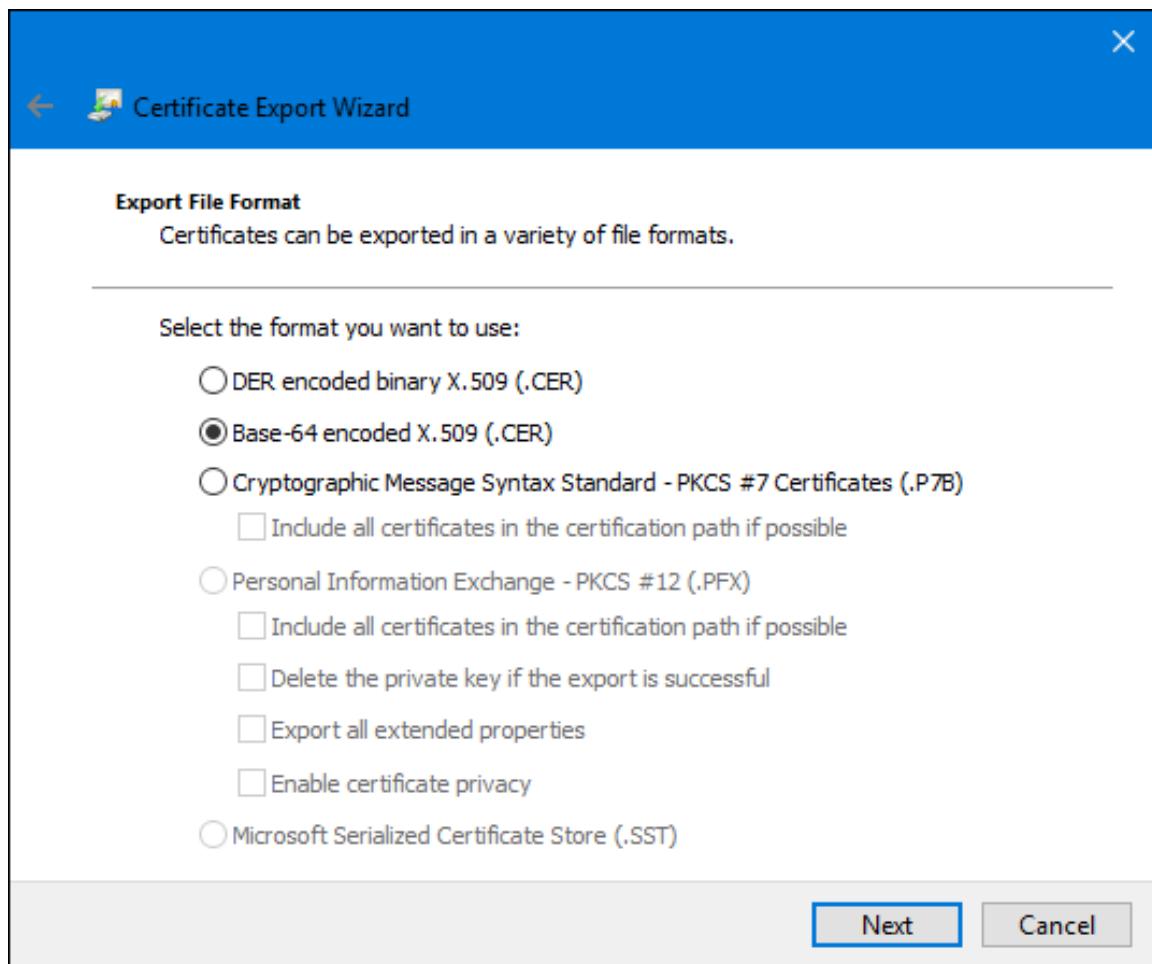


5. Close the **Group Policy Management Editor** and the **Group Policy Management** console.

#### 16.4.3. Step 3: Extracting the CA Certificate from AD

Extract the root certificate authority (CA) certificate and copy it to the Directory Server host:

- If your AD CA certificate is self-signed:
  1. On an AD DC with the **Certification Authority** application installed, press the **Super key+R** combination to open the **Run** dialog.
  2. Enter the **certsrv.msc** command and click **OK** to open the **Certification Authority** application.
  3. Right-click on the name of the local Certificate Authority and choose **Properties**.
  4. On the **General** tab, select the certificate to export in the **CA certificates** field and click **View Certificate**.
  5. On the **Details** tab, click **Copy to File** to start the **Certificate Export Wizard**.
  6. Click **Next**, and then select **Base-64 encoded X.509 (.CER)**.



7. Specify a suitable directory and file name for the exported file. Click **Next** to export the certificate, and then click **Finish**.
8. Copy the root CA certificate to the Directory Server host.
- If your AD CA certificate is signed by an external CA:
  1. Determine the root CA. For example:

```
# openssl s_client -connect adserver.example.com:636
CONNECTED(00000003)
depth=1 C = US, O = Demo Company, OU = IT, CN = Demo CA-28
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
0 s:/C=US/O=Demo Company/OU=IT/CN=adserver.example.com
 i:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
1 s:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
 i:/C=US/O=Demo Company/OU=IT/CN=Demo Root CA 2
```

The previous example shows that the AD server's CA certificate is signed by **CN=Demo CA-1**, which is signed by **CN=Demo Root CA 2**. This means that **CN=Demo Root CA 2** is the root CA.

2. Contact the operator of the root CA about how to retrieve the CA certificate.
3. Copy the root CA certificate to the Directory Server host.

#### 16.4.4. Step 4: Extracting the CA Certificate from the Directory Server's NSS Database

To extract the CA certificate from the Directory Server's NSS database:

1. List the certificates in the database:

```
# certutil -d /etc/dirsrv/slappd-instance_name -L

Certificate Nickname          Trust Attributes
SSL,S/MIME,JAR/XPI

Server-Cert                  u,u,u
Example CA                   C,,
```

2. Extract the CA certificate from the database. For example, to extract the CA certificate with the **Example CA** nickname and store it in the **/root/ds-ca.crt** file:

```
# certutil -d /etc/dirsrv/slappd-instance_name -L -n "Example CA" -a > /root/ds-ca.crt
```

3. Copy the CA certificate to the AD DC.

#### 16.4.5. Step 5: Creating the Synchronization Accounts

For synchronization between AD and Directory Server, you require one account in AD and one in Directory Server. This section explains further details about creating these accounts.

##### Creating an Account in Directory Server

The AD DCs use a Directory Server account in the **Password Sync** service to synchronize passwords to Directory Server. For example, to create the **cn=pw\_sync\_user,dc=config** user in Directory Server:

1. Create the user account:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=pw_sync_user,cn=config
objectClass: inetorgperson
objectClass: person
objectClass: top
cn: pw_sync_user
sn: pw_sync_user
userPassword: password
passwordExpirationTime: 20380101000000Z
```

This creates the **cn=pw\_sync\_user,dc=config** account and sets its expiration time to January 01 2038.



##### IMPORTANT

For security reasons, do not create the account in the synchronized subtree.

- Set an ACL at the top of the subtree that will be synchronized and grants **write** and **compare** permissions to the **cn=pw\_sync\_user,dc=config** user. For example, to add such an ACL to the **ou=People,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
  
dn: ou=People,dc=example,dc=com  
changetype: modify  
add: aci  
aci: (targetattr="userPassword")(version 3.0;acl "Password synchronization";  
allow (write,compare) userdn="ldap:///cn=pw_sync_user,dc=config";)
```

- Configure that Directory Server can store passwords in clear text in the changelog:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-  
unhashed-pw-switch=on
```

Because Directory Server uses a different password encryption than Active Directory, Directory Server must send the password in clear text to the Windows server. However, the clear text password is sent over a TLS encrypted connection that is required for password synchronization and is, therefore, not exposed to the network.

### Creating an Account in AD

To send and receive updates, Directory Server uses an AD account when connecting to AD. This account must be a member of the **Domain Admins** group or have equivalent permissions in AD. For details about creating AD accounts, see your AD documentation.

#### 16.4.6. Step 6: Installing the Password Sync Service

Install the **Password Sync** on every writable DC in your AD. For details about installing the **Password Sync** service, see the *Installing the password synchronization service* section in the [Red Hat Directory Server Installation Guide](#).

For a list of operating systems running the **Password Sync** service that Red Hat supports, see the [Red Hat Directory Server Release Notes](#).

#### 16.4.7. Step 7: Adding the CA Certificate Directory Server uses to the Password Sync Service's Certificate Database

On every DC that has the **Password Sync** service installed, add the CA certificate Directory Server uses to the **Password Sync** service's certificate database:

- Change into the **C:\Program Files\Red Hat Directory Password Synchronization\** directory:

```
> cd "C:\Program Files\Red Hat Directory Password Synchronization\"
```

- Create the certificate databases in the current directory:

```
> certutil.exe -d . -N
```

The **certutil.exe** utility prompts to set a password to the new database it creates.

- Import the CA certificate used by the Directory Server instance. You copied this certificate in [Section 16.4.4, "Step 4: Extracting the CA Certificate from the Directory Server's NSS](#)

**Database**" to the Windows DC. For example, to import the certificate from the **C:\ds-ca.crt** file and store it in the database with the **Example CA** nickname:

```
> certutil.exe -d . -A -n "Example CA" -t CT,, -a -i "C:\ds-ca.crt"
```

4. Optionally, verify that the CA certificate was stored correctly in the database:

```
> certutil.exe -d . -L
```

Certificate Nickname	Trust Attributes
	SSL,S/MIME,JAR/XPI
Example CA	CT,,

5. Reboot the Windows DC. The **Password Sync** service is not available until you reboot the system.



#### NOTE

If any AD user accounts exist when you install **Password Sync**, the service cannot synchronize the passwords for those accounts until the passwords are changed. This happens because **Password Sync** cannot decrypt a password once it has been stored in Active Directory. For details about enforcing a password reset for AD users, see the Active Directory documentation.

#### 16.4.8. Step 8: Adding the CA Certificate AD uses to Directory Server's Certificate Database

On the Directory Server host, add the CA certificate AD uses to the certificate database:

1. Import the CA certificate AD uses. You copied this certificate in [Section 16.4.3, "Step 3: Extracting the CA Certificate from AD"](#) to the Directory Server host. For example, to import the certificate from the **/root/ad-ca.crt** file and store it in the database with the **Example CA** nickname:

```
> certutil -d /etc/dirsrv/slappd-instance_name/ -A -n "Example CA" -t CT,, -a -i /root/ad-ca.crt
```

2. Optionally, verify that the CA certificate was stored correctly in the database:

```
> certutil -d /etc/dirsrv/slappd-instance_name/ -L
```

Certificate Nickname	Trust Attributes
	SSL,S/MIME,JAR/XPI
...	
Example CA	CT,,

#### 16.4.9. Step 9: Configuring the Database for Synchronization and Creating the Synchronization Agreement

This section describes how to configure the database for synchronization and create the synchronization agreement.

### 16.4.9.1. Configuring the Database for Synchronization and Creating the Synchronization Agreement Using the Command Line

The following example assumes that you have Directory Server running on a host named **ds.example.com** and an AD DC running on a host named **win-server.ad.example.com**. The following procedure describes how to configure synchronization between these hosts:

1. Enable replication for the suffix:

```
# dsconf -D "cn=Directory Manager" ldap://ds.example.com replication \
enable --suffix="dc=example,dc=com" --role="supplier" --replica-id=1
```

This command configures the **ds.example.com** host as a supplier for the **dc=example,dc=com** suffix and sets the replica ID for this entry to **1**.



#### IMPORTANT

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

2. Add the synchronization agreement and initialize the agreement. For example:

```
# dsconf -D "cn=Directory Manager" ldap://ds.example.com repl-winsync-agmt \
create --suffix="dc=example,dc=com" --host="win-server.ad.example.com" --port=636 \
--conn-protocol="LDAPS" --bind-dn="cn=user_name,cn=Users,dc=ad,dc=example,dc=com" \
--bind-passwd="password" --win-subtree="cn=Users,dc=example,dc=com" \
--ds-subtree="ou=People,dc=example,dc=com" --win-domain="AD" \
--init example-agreement
```

This command creates a replication agreement named *example-agreement*. The replication agreement defines settings, such as AD DC's host name, protocol, and authentication information, Directory Server uses when connecting and synchronizing data to the DC.

After the agreement is created, Directory Server initializes the agreement. To initialize the agreement later, omit the **--init** option. Note that synchronization does not start before you initialized the agreement. For details about initializing a synchronization agreement, see [Section 16.11.2.1, “Performing a Full Synchronization Using the Command Line”](#).

Optionally, pass the **--sync-users="on"** and **--sync-groups="on"** option to the command to automatically synchronize new Windows users and groups to Directory Server.

For further details about the options used in the command, enter:

```
# dsconf -D "cn=Directory Manager" ldap://ds.example.com repl-agmt --help
```

3. Verify that the initialization was successful:

```
# dsconf -D "cn=Directory Manager" ldap://ds.example.com repl-winsync-agmt \
init-status --suffix="dc=example,dc=com" example-agreement
Agreement successfully initialized.
```

### 16.4.9.2. Configuring the Database for Synchronization and Creating the Synchronization Agreement Using the Web Console

The following example assumes that you have Directory Server running on a host named **ds.example.com** and an AD DC running on a host named **win-server.ad.example.com**. The following procedure describes how to configure synchronization between these hosts:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Enable replication for the suffix:
  - a. Open the **Replication** menu.
  - b. Select the **dc=example,dc=com** suffix, and click **Enable Replication**.
  - c. Select **Supplier** in the **Replication Role** field and enter a replica ID. For example:

**Enable Replication**

Choose the replication role for this suffix. If it is a Master replica then you must pick a unique ID to identify it among the other Master replicas in your environment. The replication changelog will also automatically be created if it does not exist.

Replication Role	<b>Supplier</b>
Replica ID	1

You can optionally define the authentication information for this replicated suffix. Either a Manager DN and Password, a Bind Group DN, or both, can be provided. The Manager DN should be an entry under "cn=config" and if it does not exist it will be created, while the Bind Group DN is usually an existing group located in the database suffix. Typically, just the Manager DN and Password are used when enabling replication for a suffix.

Replication Manager DN	
Password	
Confirm Password	
Bind Group DN	

These settings configure the **ds.example.com** host as a supplier for the **dc=example,dc=com** suffix and sets the replica ID for this entry to **1**.

**IMPORTANT**

The replica ID must be a unique integer between **1** and **65534** for a suffix across all suppliers in the topology.

- d. Click **Enable Replication**.
4. Add the synchronization agreement and initialize agreement:
  - a. Open the **Replication** menu and select the **Winsync Agreements** entry.
  - b. Click **Create Agreement** and fill the fields. For example:

## Create Winsync Agreement

Agreement Name	example-agreement
Windows AD Host	win-server.example.com
Windows AD Port	636
Bind DN	cn=user_name,cn=Users,dc=ad,dc=example,dc=com
Bind Password	*****
Confirm Password	*****
Windows Domain Name	ad.example.com
Windows Subtree	cn=Users,dc=ad,dc=example,dc=com
DS Subtree	ou=People,dc=example,dc=com
Consumer Initialization	<b>Do Online Initialization</b>
<b>▼ Hide Advanced Settings</b>	
Connection Protocol	LDAPS
Synchronization Direction	both
Synchronization Interval	
Exclude Attributes	Start typing an attribute...
<input checked="" type="checkbox"/> Synchronize New Windows Groups <input checked="" type="checkbox"/> Synchronize New Windows Users <input checked="" type="checkbox"/> Keep Replication In Constant Synchronization	
<a href="#">Cancel</a> <a href="#">Save Agreement</a>	

These settings will create a synchronization agreement named **example-agreement**. The synchronization agreement defines settings, such as the DC's host name, protocol, and authentication information, Directory Server uses when connecting and synchronizing data.

Optionally, select **Sync New Windows Users** and **Sync New Windows Groups** to automatically synchronize new Windows users and groups to Directory Server.

After the agreement is created, Directory Server initializes the agreement. To initialize the agreement later, do not select **Do Online Initialization**. Note that synchronization does not start before you initialized the agreement. For details about initializing a synchronization agreement, see [Section 16.11.2.2, “Performing a Full Synchronization Using the Web Console”](#).

- c. Click **Save Agreement**.
5. Verify that the initialization was successful:
  - a. Open the **Replication** menu.
  - b. Select the **Agreements** entry.

If the initialization completed successfully, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
<b>Error (0) Replica acquired successfully: Incremental update succeeded</b>	<i>Initialized</i>

Depending of the amount of data to synchronize, the initialization can take up to several hours.

## 16.5. SYNCHRONIZING USERS

Users are not automatically synchronized between Directory Server and Active Directory. Synchronization both directions has to be configured:

- Users in the Active Directory domain are synchronized if it is configured in the sync agreement by selecting the **Sync New Windows Users** option. All of the Windows users are copied to the Directory Server when synchronization is initiated and then new users are synchronized over when they are created.
- A Directory Server user account is synchronized to Active Directory through specific attributes that are present on the Directory Server entry. Any Directory Server entry must have the **ntUser** object class and the **ntUserCreateNewAccount** attribute; the **ntUserCreateNewAccount** attribute (even on an existing entry) signals the Directory Server Windows Synchronization plug-in to write the entry over to the Active Directory server.

New or modified user entries with the **ntUser** object class added are created and synchronized over to the Windows machine at the next regular update, which is a standard poll of entry.



## NOTE

A user is not active on the Active Directory domain until it has a password. When an existing user is modified to have the required Windows attributes, that user entry will be synchronized over to the Active Directory domain, but will not be able to log in until the password is changed on the Directory Server side or an administrator sets the password on Active Directory. This is because passwords stored in the Directory Server are encrypted, and **Password Sync** cannot sync already encrypted passwords.

To make the user active on the Active Directory domain, reset the user's password.

All synchronized entries *in the Directory Server*, whether they originated in the Directory Server or in Active Directory, have special synchronization attributes:

- *ntUserDomainId*. This corresponds to the **sAMAccountName** attribute for Active Directory entries.
- *ntUniqueId*. This contains the value of the **objectGUID** attribute for the corresponding Windows entry. This attribute is set by the synchronization process and should not be set or modified manually.
- *ntUserDeleteAccount*. This attribute is set automatically when a Windows entry is synchronized over but must be set manually for Directory Server entries. If **ntUserDeleteAccount** has the value **true**, the corresponding Windows entry be deleted when the Directory Server entry is deleted. Otherwise, the entry remains in Active Directory, but is removed from the Directory Server database if it is deleted in the Directory Server.

Setting **ntUserCreateNewAccount** and **ntUserDeleteAccount** on Directory Server entries allows the Directory Manager precise control over which users within the synchronized subtree are synchronized on Active Directory.

### 16.5.1. User Attributes Synchronized between Directory Server and Active Directory

Only a subset of Directory Server and Active Directory attributes are synchronized. These attributes are hard-coded and are defined regardless of which way the entry is being synchronized. Any other attributes present in the entry, either in Directory Server or in Active Directory, remain unaffected by synchronization.

Some attributes used in Directory Server and Active Directory are identical. These are usually attributes defined in an LDAP standard, which are common among all LDAP services. These attributes are synchronized to one another exactly. [Table 16.2, "User Schema That Are the Same in Directory Server and Windows Servers"](#) shows attributes that are the same between the Directory Server and Windows servers.

Some attributes define the same information, but the names of the attributes or their schema definitions are different. These attributes are mapped between Active Directory and Directory Server, so that attribute A in one server is treated as attribute B in the other. For synchronization, many of these attributes relate to Windows-specific information. [Table 16.1, "User Schema Mapped between Directory Server and Active Directory"](#) shows the attributes that are mapped between the Directory Server and Windows servers.

For more information on the differences in ways that Directory Server and Active Directory handle some schema elements, see [Section 16.5.2, "User Schema Differences between Red Hat Directory Server and Active Directory"](#).

**Table 16.1. User Schema Mapped between Directory Server and Active Directory**

Directory Server	Active Directory
cn[a]	name
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

[a] The **cn** is treated differently than other synchronized attributes. It is mapped directly (**cn** to **cn**) when syncing from Directory Server to Active Directory. When syncing from Active Directory to Directory Server, however, **cn** is mapped from the **name** attribute on Windows to the **cn** attribute in Directory Server.

**Table 16.2. User Schema That Are the Same in Directory Server and Windows Servers**

cn[a]	physicalDeliveryOfficeName
description	postOfficeBox
destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenname	registeredAddress
homePhone	sn

homePostalAddress	st
initials	street
I	telephoneNumber
mail	teletexTerminalIdentifier
mobile	telexNumber
o	title
ou	usercertificate
pager	x121Address

[a] The **cn** is treated differently than other synchronized attributes. It is mapped directly (**cn** to **cn**) when syncing from Directory Server to Active Directory. When syncing from Active Directory to Directory Server, however, **cn** is mapped from the **name** attribute on Windows to the **cn** attribute in Directory Server.

## 16.5.2. User Schema Differences between Red Hat Directory Server and Active Directory

Although Active Directory supports the same basic X.500 object classes as Directory Server, there are a few incompatibilities of which administrators should be aware.

### 16.5.2.1. Values for cn Attributes

In Directory Server, the **cn** attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the Directory Server **cn** attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that, potentially, if a **cn** value is added to an Active Directory entry and that value is not one of the values for **cn** in Directory Server, then all of the Directory Server **cn** values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the **cn** attribute as its naming attribute, where Directory Server uses **uid**. This means that there is the potential to rename the entry entirely (and accidentally) if the **cn** attribute is edited in the Directory Server. If that **cn** change is written over to the Active Directory entry, then the entry is renamed, and the new named entry is written back over to Directory Server.

### 16.5.2.2. Password Policies

Both Active Directory and Directory Server can enforce password policies such as password minimum length or maximum age. Windows Synchronization makes no attempt to ensure that the policies are consistent, enforced, or synchronized. If password policy is not consistent in both Directory Server and Active Directory, then password changes made on one system may fail when synchronized to the other system. The default password syntax setting on Directory Server mimics the default password complexity rules that Active Directory enforces.

### 16.5.2.3. Values for street and streetAddress

Active Directory uses the attribute ***streetAddress*** for a user or group's postal address; this is the way that Directory Server uses the ***street*** attribute. There are two important differences in the way that Active Directory and Directory Server use the ***streetAddress*** and ***street*** attributes, respectively:

- In Directory Server, ***streetAddress*** is an alias for ***street***. Active Directory also has the ***street*** attribute, but it is a separate attribute that can hold an independent value, not an alias for ***streetAddress***.
- Active Directory defines both ***streetAddress*** and ***street*** as single-valued attributes, while Directory Server defines ***street*** as a multi-valued attribute, as specified in RFC 4519.

Because of the different ways that Directory Server and Active Directory handle ***streetAddress*** and ***street*** attributes, there are two rules to follow when setting address attributes in Active Directory and Directory Server:

- Windows Synchronization maps ***streetAddress*** in the Windows entry to ***street*** in Directory Server. To avoid conflicts, the ***street*** attribute should not be used in Active Directory.
- Only one Directory Server ***street*** attribute value is synchronized to Active Directory. If the ***streetAddress*** attribute is changed in Active Directory and the new value does not already exist in Directory Server, then all ***street*** attribute values in Directory Server are replaced with the new, single Active Directory value.

### 16.5.2.4. Constraints on the initials Attribute

For the ***initials*** attribute, Active Directory imposes a maximum length constraint of six characters, but Directory Server does not have a length limit. If an ***initials*** attribute longer than six characters is added to Directory Server, the value is trimmed when it is synchronized with the Active Directory entry.

### 16.5.3. Configuring User Synchronization for Directory Server Users

For Directory Server users to be synchronized over to Active Directory, the user entries must have the appropriate sync attributes set.

To enable synchronization through the command line, add the required sync attributes to an entry or create an entry with those attributes.

Three schema elements are required for synchronization:

- The ***ntUser*** object class
- The ***ntUserDomainId*** attribute, to give the Windows ID
- The ***ntUserCreateNewAccount*** attribute, to signal to the synchronization plug-in to sync the Directory Server entry over to Active Directory

For example, using the ***ldapmodify*** utility:

```
dn: uid=scarter,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass:ntUser
-
add: ntUserDomainId
```

```

ntUserDomainId: Sam Carter
-
add: ntUserCreateNewAccount
ntUserCreateNewAccount: true
-
add: ntUserDeleteAccount
ntUserDeleteAccount: true

```

Many additional Windows and user attributes can be added to the entry. All of the schema which is synchronized is listed in [Section 16.5.1, "User Attributes Synchronized between Directory Server and Active Directory"](#). Windows-specific attributes, belonging to the **ntUser** object class, are described in more detail in the [Red Hat Directory Server 11 Configuration, Command, and File Reference](#).



#### NOTE

Reset the user's password.

A user is not active on the Active Directory domain until it has a password. When an existing user is modified to have the required Windows attributes, that user entry will be synchronized over to the Active Directory domain, but will not be able to log in until the password is changed on the Directory Server side or an administrator sets the password on Active Directory. **Password Sync** cannot sync encrypted passwords.

So, to make the user active on the Active Directory domain, reset the user's password.

#### 16.5.4. Configuring User Synchronization for Active Directory Users

Synchronization for Windows users (users which originate in the Active Directory domain) is configured in the sync agreement.

To enable user synchronization:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt set --sync-users="on" --suffix="dc=example,dc=com" example-agreement
```

To disable user synchronization, set the **--sync-users** option to **off**.

## 16.6. SYNCHRONIZING GROUPS

Like user entries, groups are not automatically synchronized between Directory Server and Active Directory. Synchronization both directions has to be configured:

- Groups in the Active Directory domain are synchronized if it is configured in the sync agreement by selecting the **Sync New Windows Groups** option. All of the Windows groups are copied to the Directory Server when synchronization is initiated and then new groups are synchronized over as they are created.
- A Directory Server group account is synchronized to Active Directory through specific attributes that are present on the Directory Server entry. Any Directory Server entry must have the **ntGroup** object class and the **ntGroupCreateNewGroup** attribute; the **ntGroupCreateNewGroup** attribute (even on an existing entry) signals Directory Server Windows Synchronization to write the entry over to the Active Directory server.

New or modified groups that have the **ntGroup** object class are created and synchronized over to the Windows machine at the next regular update.



## IMPORTANT

When a group is synchronized, the list of all of its members is also synchronized. However, the member entries themselves are not synchronized unless user sync is enabled and applies to those entries.

This could create a problem when an application or service tries to do a modify operation on all members in a group on the Active Directory server, if some of those users do not exist.

Additionally, groups have a few other common attributes:

- Two attributes control whether Directory Server groups are created and deleted on Active Directory, ***ntGroupCreateNewGroup*** and ***ntGroupDeleteGroup***.  
***ntGroupCreateNewGroup*** is required to sync Directory Server groups over to Active Directory.
- *ntUserDomainId* contains the unique ID for the entry on the Active Directory domain. This is the only required attribute for the **ntGroup** object class.
- *ntGroupType* is the type of Windows group. Windows group types are global/security, domain local/security, builtin, universal/security, global/distribution, domain local/distribution, or universal/distribution. This is set automatically for Windows groups that are synchronized over, but this attribute must be set manually on Directory Server entries before they can be synchronized.

### 16.6.1. About Windows Group Types

In Active Directory, there are two major types of groups: security and distribution. Security groups are most similar to groups in Directory Server, since security groups can have policies configured for access controls, resource restrictions, and other permissions. Distribution groups are for mailing distribution. These are further broken down into global and local groups. The Directory Server *ntGroupType* supports all four group types:

- **-2147483646** for global/security (the default)
- **-2147483644** for domain local/security
- **-2147483643** for builtin
- **-2147483640** for universal/security
- **2** for global/distribution
- **4** for domain local/distribution
- **8** for universal/distribution

### 16.6.2. Group Attributes Synchronized between Directory Server and Active Directory

Only a subset of Directory Server and Active Directory attributes are synchronized. These attributes are hard-coded and are defined regardless of which way the entry is being synchronized. Any other attributes present in the entry, either in Directory Server or in Active Directory, remain unaffected by synchronization.

Some attributes used in Directory Server and Active Directory group entries are identical. These are usually attributes defined in an LDAP standard, which are common among all LDAP services. These attributes are synchronized to one another exactly. [Table 16.4, "Group Entry Attributes That Are the Same between Directory Server and Active Directory"](#) shows attributes that are the same between the Directory Server and Windows servers.

Some attributes define the same information, but the names of the attributes or their schema definitions are different. These attributes are mapped between Active Directory and Directory Server, so that attribute A in one server is treated as attribute B in the other. For synchronization, many of these attributes relate to Windows-specific information. [Table 16.3, "Group Entry Attribute Mapping between Directory Server and Active Directory"](#) shows the attributes that are mapped between the Directory Server and Windows servers.

For more information on the differences in ways that Directory Server and Active Directory handle some schema elements, see [Section 16.6.3, "Group Schema Differences between Red Hat Directory Server and Active Directory"](#).

**Table 16.3. Group Entry Attribute Mapping between Directory Server and Active Directory**

Directory Server	Active Directory
cn	name
ntUserDomainID	name
ntGroupType	groupType
uniqueMember	Member[a]
member	

[a] The **Member** attribute in Active Directory is synchronized to the **uniqueMember** attribute in Directory Server.

**Table 16.4. Group Entry Attributes That Are the Same between Directory Server and Active Directory**

cn	o
description	ou
l	seeAlso
mail	

### 16.6.3. Group Schema Differences between Red Hat Directory Server and Active Directory

Although Active Directory supports the same basic X.500 object classes as Directory Server, there are a few incompatibilities of which administrators should be aware.

Nested groups (where a group contains another group as a member) are supported and for Windows Synchronization are synchronized. However, Active Directory imposes certain constraints as to the composition of nested groups. For example, a global group is not allowed to contain a domain local group as a member. Directory Server has no concept of local and global groups, and, therefore, it is possible to create entries on the Directory Server side that violate Active Directory's constraints when synchronized.

#### 16.6.4. Configuring Group Synchronization for Directory Server Groups

For Directory Server groups to be synchronized over to Active Directory, the group entries must have the appropriate sync attributes set.

To enable synchronization through the command line, add the required sync attributes to an entry or create an entry with those attributes.

Three schema elements are required for synchronization:

- The **ntGroup** object class.
- The **ntUserDomainId** attribute, to give the Windows ID for the entry.
- The **ntGroupCreateNewGroup** attribute, to signal to the synchronization plug-in to sync the Directory Server entry over to Active Directory.

The **ntGroupDeleteGroup** attribute is optional, but this sets whether to delete the entry automatically from the Active Directory domain if it is deleted in the Directory Server.

It is also recommended to add the **ntGroupType** attribute. If this attribute is not specified, then the group is automatically added as a global security group (**ntGroupType:-2147483646**).

For example, using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
  
dn: cn=Example Group,ou=Groups,dc=example,dc=com  
changetype: modify  
add: objectClass  
objectClass:ntGroup  
  
-  
add: ntUserDomainId  
ntUserDomainId: example-group  
  
-  
add: ntGroupCreateNewGroup  
ntGroupCreateNewGroup: true  
  
-  
add: ntGroupDeleteGroup  
ntGroupDeleteGroup: true  
  
-  
add: ntGroupType  
ntGroupType: 2
```

Many additional Windows and group attributes can be added to the entry. All of the schema which is synchronized is listed in [Section 16.6.2, “Group Attributes Synchronized between Directory Server and Active Directory”](#). Windows-specific attributes, belonging to the **ntGroup** object class, are described in more detail in the [Red Hat Directory Server 11 Configuration, Command, and File Reference](#).

## 16.6.5. Configuring Group Synchronization for Active Directory Groups

Synchronization for Windows users (users which originate in the Active Directory domain) is configured in the sync agreement.

To enable group synchronization:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt set --sync-groups="on" --suffix="dc=example,dc=com" example-agreement
```

To disable group synchronization, set the **--sync-groups** option to **off**.

## 16.7. CONFIGURING UNI-DIRECTIONAL SYNCHRONIZATION

As [Figure 16.1, “Active Directory – Directory Server Synchronization Process”](#) illustrates, synchronization is *bi-directional* by default. That means that changes in Active Directory are sent to Directory Server and changes on Directory Server are sent to Active Directory.

It is possible to create *uni-directional* synchronization, where changes are only sent one-way. This is similar to a supplier-consumer relationship<sup>[1]</sup> as opposed to multi-supplier.

An additional attribute for the sync agreement, ***oneWaySync***, enables uni-directional synchronization and specifies the direction to send changes. The possible values are ***fromWindows*** (for Active Directory to Directory Server sync) and ***toWindows*** (for Directory Server to Active Directory sync). If this attribute is absent, then synchronization is bi-directional.

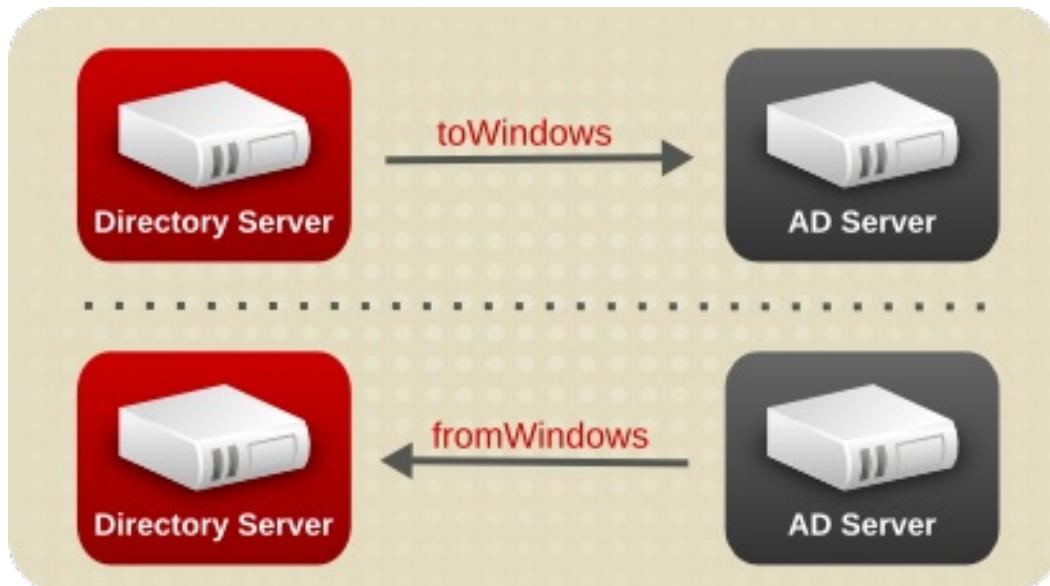


Figure 16.3. Uni-Directional Synchronization

The synchronization process itself is the mostly same for bi-directional and uni-directional synchronization. It uses the same sync interval and configuration. The only difference is in how sync information is requested.

For Windows Active Directory to Directory Server synchronization, during the regular synchronization update interval, the Directory Server contacts the Active Directory server and sends the DirSync control to request updates. However, the Directory Server does not send any changes or entries from its side. So, the sync update consists of the Active Directory changes being sent to and updating the Directory Server entries.

For Directory Server to Active Directory synchronization, the Directory Server sends entry modifications to the Active Directory server in a normal update, but it does not include the DirSync control so that it does not request any updates from the Active Directory side.

Use the **--one-way-sync="*direction*"** option to enable uni-directional synchronization in one of the following situations:

1. If you create a new synchronization agreement in [Section 16.4.9, "Step 9: Configuring the Database for Synchronization and Creating the Synchronization Agreement"](#), pass the option to the **dsconf repl-winsync-agmt create** command.
2. If the synchronization agreement already exists, update the agreement. For example, to set synchronization from AD to Directory Server:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt set --one-way-sync="fromWindows" --suffix="dc=example,dc=com" example-agreement
```



#### NOTE

Enabling uni-directional sync does *not* automatically prevent changes on the unsynchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync is configured to go from Active Directory to Directory Server, so Active Directory is (in essence) the data supplier. If an entry is modified or even deleted on the Directory Server, then the Directory Server information is different than the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

To prevent data inconsistency, use access control rules to prevent editing or deleting entries within the synchronized subtree on the unsynchronized server. Access controls for Directory Server are covered in [Chapter 18, Managing Access Control](#). For Active Directory, see the appropriate Windows documentation.

Uni-directional sync does not affect password synchronization. Even when the synchronization direction is set to **toWindows**, after updating a password on the Active Directory server, the password is sent to the Directory Server.

## 16.8. CONFIGURING MULTIPLE SUBTREES AND FILTERS IN WINDOWS SYNCHRONIZATION

Windows Synchronization is designed to synchronize between multiple pairs of subtrees on the Directory Server (DS) and Active Directory (AD). By using filters, only specified entries under a subtree are synchronized.

### Multiple Subtrees in Windows Synchronization

To synchronize among multiple subtree pairs, configure the Directory Server and the Active Directory subtrees in the **winSyncSubtreePair** parameter in the Windows sync agreement. For example to set multiple the **ou=OU1,dc=DSexample,dc=com** and **ou=OU1,DC=ADexample,DC=com** subtree:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt set --subtree-pair="ou=OU1,dc=DSexample,dc=com:ou=OU1,DC=ADexample,DC=com" --suffix="dc=example,dc=com" example-agreement
```

If **winSyncSubtreePair** is not set, the **nsds7WindowsReplicaSubtree** AD subtree parameter and the **nsds7DirectoryReplicaSubtree** DS subtree parameter are used for the synchronization target checks instead. Otherwise, these two parameters are ignored.

## Filters in Windows Synchronization

You can set a filter that selects data to be synchronized in the following parameters:

- **--win-filter** sets an additional filter on the Active Directory server,
- **--ds-filter** parameter sets an additional filter on Directory Server.

The following example configures that the **example\_agreement** synchronizes entries that contain **user** and **group** attributes:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt \
    set --win-filter="(|(cn=*user*)(cn=*group*))" --ds-filter="(|(uid=*user*)(cn=*group*))" \
    example_agreement
```

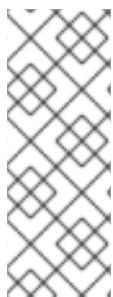
## 16.9. SYNCHRONIZING POSIX ATTRIBUTES FOR USERS AND GROUPS

A subset of all possible user and attributes are synchronized between Active Directory and Red Hat Directory Server. Some attributes are mapped, where there are differences between Active Directory and Directory Server schemas, and some attributes are matched directly. The attributes (matched and mapped) which are synchronized are listed in [Section 16.5.1, “User Attributes Synchronized between Directory Server and Active Directory”](#) and [Section 16.6.2, “Group Attributes Synchronized between Directory Server and Active Directory”](#).

By default, only those attributes are synchronized.

One type of attribute that is missing from that sync list is any POSIX-related attribute. On Linux systems, system users and groups are identified as POSIX entries, and LDAP POSIX attributes contain that required information. However, when Windows users are synchronized over, they have **ntUser** and **ntGroup** attributes automatically added which identify them as Windows accounts, but no POSIX attributes are synchronized over (even if they exist on the Active Directory entry) and no POSIX attributes are added on the Directory Server side.

The Posix Winsync API Plug-in synchronizes POSIX attributes between Active Directory and Directory Server entries.



### NOTE

All POSIX attributes (such as **uidNumber**, **gidNumber**, and **homeDirectory**) are synchronized between Active Directory and Directory Server entries. However, if a new POSIX entry or POSIX attributes are added to an existing entry in the Directory Server, *only the POSIX attributes are synchronized over to the Active Directory corresponding entry*. The POSIX object class (**posixAccount** for users and **posixGroup** for groups) is not added to the Active Directory entry.

### 16.9.1. Enabling POSIX Attribute Synchronization

The **Posix Winsync API** Plug-in is disabled by default and must be enabled for POSIX attributes to be synchronized from Active Directory user and group entries to the corresponding Directory Server entries.

To enable the **Posix Winsync API** plug-in:

1. Enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin "cn=Posix Winsync API,cn=plugins,cn=config" enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

If you enable the dynamic plug-in as described in [Section 1.10.2, “Enabling Plug-ins Dynamically”](#), restarting the instance is not required.

### 16.9.2. Changing Posix Group Attribute Synchronization Settings

There are multiple plug-in attributes that can be set to control how the POSIX group attributes and group members are synchronized from the Active Directory entry to the corresponding Directory Server group and user entries. For details, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

The defaults can be used for most deployments, but the settings can be changed depending on the Active Directory environment. For example, to enable nested group mappings:

1. Use the following command to enable the nested group mapping:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin posix-winsync set --map-nested-grouping="true"
```

2. Restart the Directory Server to load the new configuration.

```
# dsctl instance_name restart
```

### 16.9.3. Fixing Mismatched member and uniqueMember Attribute Values in posixGroup Entries

If the **member** and **uniqueMember** attribute values in **posixGroup** entries on Directory Server and Active Directory (AD) do not match, use the **dsconf plugin posix-winsync fixup** command to fix the problem:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin posix-winsync fixup DN
```

This command recreates **memberUid** values on Directory Server and automatically modifies the **member** and **uniqueMember** attribute values to match the values defined in AD.

Optionally, pass the **-f filter** parameter to the command to specify in which entries the command should fix **memberUid** attributes. Without a filter, the command operates on all entries that contain the **inetuser**, **inetadmin**, and **nsmemberof** object class.

## 16.10. DELETING AND RESURRECTING ENTRIES

This section describes how enabling synchronization affects deleted entries on the sync peers and how resurrected entries are handled.

### 16.10.1. Deleting Entries

All changes on an Active Directory peers are always synchronized back to the Directory Server. This means that when an Active Directory group or user account is deleted on the Active Directory domain, the deletion is automatically synchronized back to the Directory Server sync peer server.

On Directory Server, on the other hand, when a Directory Server account is deleted, the corresponding entry on Active Directory is only deleted if the Directory Server entry has the ***ntUserDeleteAccount*** or ***ntGroupDeleteGroup*** attribute set to **true**.



#### NOTE

When a Directory Server entry is synchronized over to Active Directory for the first time, Active Directory automatically assigns it a unique ID. At the next synchronization interval, the unique ID is synchronized back to the Directory Server entry and stored as the ***ntUniqueId*** attribute. If the Directory Server entry is deleted on Active Directory *before* the unique ID is synchronized back to Directory Server, the entry *will not* be deleted on Directory Server. Directory Server uses the ***ntUniqueId*** attribute to identify and synchronize changes made on Active Directory to the corresponding Directory Server entry; without that attribute, Directory Server will not recognize the deletion.

To delete the entry on Active Directory and then synchronize the deletion over to Directory Server, wait the length of the ***winSyncInterval*** (by default, five minutes) after the entry is created before deleting it so that the ***ntUniqueId*** attribute is synchronized.

### 16.10.2. Resurrecting Entries

It is possible to add deleted entries back in Directory Server; the deleted entries are called *tombstone* entries. When a deleted entry which was synchronized between Directory Server and Active Directory is re-added to Directory Server, the resurrected Directory Server entry has all of its original attributes and values. This is called *tombstone reanimation*. The resurrected entry includes the original ***ntUniqueId*** attribute which was used to synchronize the entries, which signals to the Active Directory server that this new entry is a tombstone entry.

Active Directory resurrects the old entry and preserves the original unique ID for the entry.

For Active Directory entries, when the tombstone entry is resurrected on Directory Server, all of the attributes of the original Directory Server are retained and are still included in the resurrected Active Directory entry.

## 16.11. SENDING SYNCHRONIZATION UPDATES

Synchronization occurs as frequently as is set in the ***winSyncInterval*** setting (for retrieving changes from the Active Directory domain) or ***nsds5replicaupdateschedule*** setting (for pushing changes from the Directory Server). By default, changes are retrieved from Active Directory every five minutes, and changes from the Directory Server are sent immediately.

A sync update can be triggered manually. It is also possible to do a full resynchronization, which sends and pulls every entry in the Directory Server and Active Directory as if it were new. A full resynchronization includes existing Directory Server entries which may not have previously been synchronized.

## 16.11.1. Performing a Manual Incremental Synchronization

During normal operations, all the updates made to entries in the Directory Server that need to be sent to Active Directory are collected to the changelog and then replayed during an incremental update.

To manually synchronize the changes:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt poke --suffix="dc=example,dc=com" example-agreement
```

## 16.11.2. Performing a Full Synchronization

If there have been major changes to data, or synchronization attributes are added to pre-existing Directory Server entries, it is necessary to initiate a *resynchronization*. Resynchronization is a total update; the entire contents of synchronized subtrees are examined and, if necessary, updated. Resynchronization is done without using the changelog. This is similar to initializing or reinitializing a consumer in replication.

### 16.11.2.1. Performing a Full Synchronization Using the Command Line

To start a full synchronization using the command line:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt init --suffix="suffix" agreement_name
```

To display the synchronization status:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt init-status --suffix="suffix" agreement_name
```

### 16.11.2.2. Performing a Full Synchronization Using the Web Console

To start a full synchronization:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Replication** menu and select the **Winsync Agreements** entry.
4. Open the **Choose Action** menu next to the synchronization agreement you want to synchronize and select **Full Re-Synchronization**.

Resynchronizing does not delete data on the sync peer. The process sends and receives all updates and adds any new or modified Directory Server entries. For example, the process adds a pre-existing Directory Server user that had the **ntUser** object class added.

To display the synchronization status in the web console:

1. Open the **Replication** menu.
2. Select the **Winsync Agreements** entry.

If the synchronization completed successfully, the web console displays the **Error (0) Replica acquired successfully: Incremental update succeeded** message in the **Last Update Status** column.

Last Update Status	Last Init Status
Error (0) Replica acquired successfully: Incremental update succeeded	Initialized

### 16.11.3. Setting Synchronization Schedules

Synchronization works two ways. The Directory Server sends its updates to Active Directory on a configurable schedule, similar to replication, using the **nsds5replicaupdateschedule** attribute. The Directory Server polls the Active Directory to check for changes; the frequency that it checks the Active Directory server is set in the **winSyncInterval** attribute.

By default, the Directory Server update schedule is to always be in sync. The Active Directory interval is to poll the Active Directory every five minutes.

To change the schedule the Directory Server uses to send its updates to the Active Directory, edit the **nsds5replicaupdateschedule** attribute. The schedule is set with start (SSSS) and end (EEEE) times in the form HHMM, using a 24-hour clock. The days to schedule sync updates are use ranging from **0** (Sunday) to **6** (Saturday).

**nsds5replicaupdateschedule: SSSS EEEE DDDDDDD**

For example, this schedules synchronization to run from noon to 2:00pm on Sunday, Tuesday, Thursday, and Saturday:

**nsds5replicaupdateschedule: 1200 1400 0246**



#### NOTE

The synchronization times cannot wrap around midnight, so the setting **2300 0100** is not valid.

To change how frequently the Directory Server checks the Active Directory for changes to Active Directory entries, reset the **winSyncInterval** attribute. This attribute is set in seconds, so the default of **300** means that the Directory Server polls the Active Directory server every 300 seconds, or five minutes. Setting this to a higher value can be useful if the directory searches are taking too long and affecting performance.

**winSyncInterval: 1000**

### 16.11.4. Changing Synchronization Connections

Two aspects of the connection for the sync agreement can be altered:

- The bind user name and password (**nsDS5ReplicaBindDN** and **nsDS5ReplicaCredentials**).

- The connection method (***nsDS5ReplicaTransportInfo***).

It is only possible to change the ***nsDS5ReplicaTransportInfo*** from **LDAP** to **StartTLS** and vice versa. It is not possible to change to or from **LDAPS** because it is not possible to change the port number, and switching between LDAP and LDAPS requires changing the port number.

For example:

```
nsDS5ReplicaBindDN: cn=sync user,cn=Users,dc=ad1
nsDS5ReplicaCredentials: {DES}ffGad646dT0nnsT8nJOaMA==
nsDS5ReplicaTransportInfo: StartTLS
```



### WARNING

It is not possible to change the port number of the Active Directory sync peer. Therefore, it is also not possible to switch between standard/STARTTLS connections and TLS connections, since that requires changing between standard and insecure ports.

To change to or from TLS, delete the sync agreement and add it again with the updated port number and new transport information.

### 16.11.5. Handling Entries That Move Out of the Synchronized Subtree

The sync agreement defines what subtrees in both Active Directory and Directory Server are synchronized between each other. Entries *within* the scope (the subtree) are synchronized; other entries are ignored.

However, the synchronization process actually starts at the root DN to begin evaluating entries for synchronization. Entries are correlated based on the ***samAccount*** in the Active Directory and the ***uid*** attribute in Directory Server. The synchronization plug-in notes if an entry (based on the ***samAccount/uid*** relationship) is removed from the synchronized subtree either because it is deleted or moved. That is the signal to the synchronization plug-in that the entry is no longer to be synchronized.

The issue is that the sync process needs some configuration to determine how to handle that moved entry. There are three options: delete the corresponding entry, ignore the entry (the default), or unsync the entry.



### NOTE

These sync actions only relate to how to handle *on the Directory Server side* when an entry is moved out of scope on the Active Directory side. This does not affect any Active Directory entry if an entry is moved out of the synchronized subtree on the Directory Server side.

The default behavior in Directory Server 9.0 was to delete the corresponding Directory Server entry. *This was true even if the entry on the Active Directory side was never synchronized over to the Directory Server side.* Starting in Directory Server 9.1, the default behavior is to ignore the entry and take no action.

For example, a user with the **samAccount** ID of **jsmith** was created in the **ou=Employees** subtree on Active Directory. The synchronized subtree is **ou=Users**, so the **jsmith** user was never synchronized over to Directory Server.

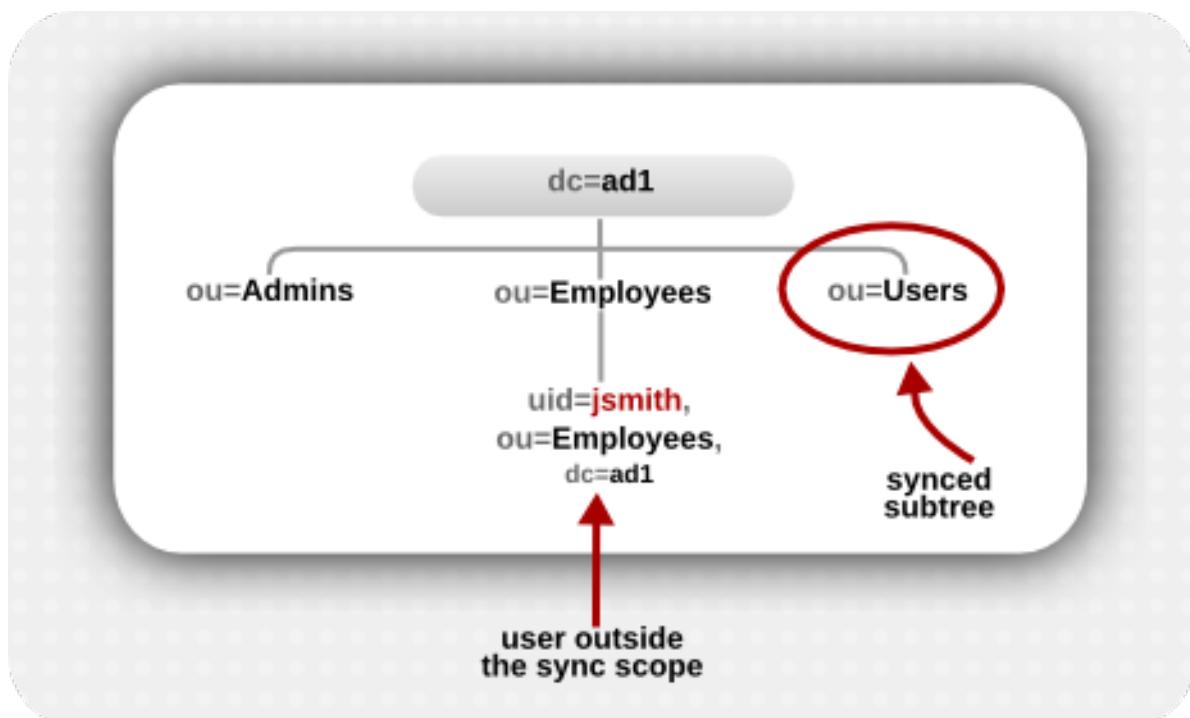


Figure 16.4. Active Directory Tree

For 7.x and 8.x versions of Directory Server, synchronization simply ignored that user, since it was outside the synchronized subtree.

Starting in Directory Server 9.0, Directory Server began supporting subtree renames – which means that existing entries could be moved between branches of the directory tree. The synchronization plugin, then, assumes that entries in the Active Directory tree which correspond to a Directory Server user (**samAccount/uid** relationship) but are outside the synchronized subtree are *intentionally* moved outside the synchronized subtree – essentially, a rename operation. The assumption then was that the "corresponding" Directory Server entry should be deleted.

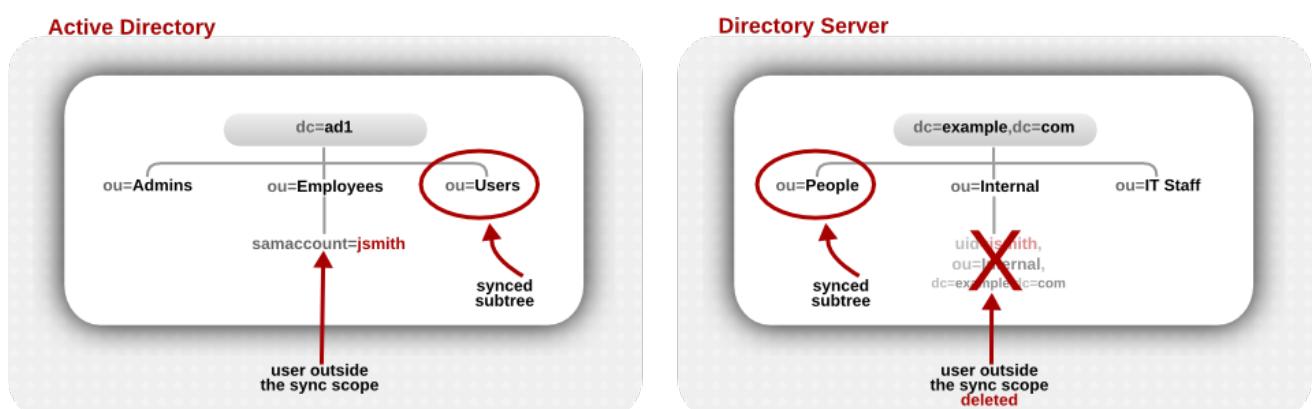


Figure 16.5. Active Directory and Directory Server Trees Compared

This assumption is not necessarily an accurate one, particularly for user entries which always existed outside the synchronized subtree.

The **winSyncMoveAction** attribute for the synchronization agreement sets instructions on how to handle these moved entries:

- **none** takes no action, so if a synchronized Directory Server entry exists, it may be synchronized over to or create an Active Directory entry *within* scope. If no synchronized Directory Server entry exists, nothing happens at all (this is the default behavior in the Directory Server version 9.1 and later).
- **unsync** removes any sync-related attributes (***ntUser*** or ***ntGroup***) from the Directory Server entry but otherwise leaves the Directory Server entry intact.



### IMPORTANT

There is a risk when unsyncing entries that the Active Directory entry may be deleted at a later time, and the Directory Server entry will be left intact. This can create data inconsistency issues, especially if the Directory Server entry is ever used to recreate the entry on the Active Directory side later.

- **delete** deletes the corresponding entry on the Directory Server side, regardless of whether it was ever synchronized with Active Directory (this was the default behavior in 9.0).



### IMPORTANT

You almost never want to delete a Directory Server entry without deleting the corresponding Active Directory entry. This option is available only for compatibility with Directory Server 9.0 systems.

If it is necessary to change the default:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-winsync-agmt --move-action="action" --suffix="suffix" agreement_name
```

## 16.12. TROUBLESHOOTING

If synchronization does not seem to function properly, see the Windows event log or Directory Server error log for information on any potential problems.

### Enable replication logging to record synchronization errors

Enable replication logging for more detailed information on synchronization to be recorded in the error logs. The replication log level produces more verbose logs from the sync code. Messages related to synchronization traffic (which is the same as replication traffic) can help in diagnosing problems.

For details about configuring log levels, see [Section 21.3.7, “Configuring the Log Levels”](#).

### Error #1: After synchronization, the status returns error 81.

One of the sync peer servers has not been properly configured for TLS communication. Examine the Directory Server access log file to see if the connection attempt was received by the Directory Server. There are also 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 (including the port number, host name or IPv4/IPv6 address, search base, and user credentials) 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 to the Directory Server is successful, it is likely that the misconfiguration is on Active Directory. Examine the Windows event log file for error messages.



## NOTE

A common problem is that the certificate authority was not configured as trusted when the Windows sync services certificate database was configured.

### Error #2: An entry is moved from one subtree on Active Directory to another subtree, but the user is not moved to the corresponding subtree on Directory Server.

This is a known issue with synchronizing modrdn operations on Active Directory with entries on Directory Server. To work around it, delete the entry on Active Directory and then add it anew to the new subtree. The deletion and the addition will be properly synchronized to the Directory Server peer.

---

[1] Unlike a consumer, changes can still be made on the un-synchronized server. Use ACLs to prevent editing or deleting entries on the un-synchronized server to maintain data integrity.

# CHAPTER 17. SETTING UP CONTENT SYNCHRONIZATION USING THE SYNCREPL PROTOCOL

Using the **Content Synchronization** plug-in, Directory Server supports the **SyncRepl** protocol according to [RFC 4533](#). This protocol enables LDAP servers and clients to use Red Hat Directory Server as a source to synchronize their local database with the changing content of Directory Server.

To use the **SyncRepl** protocol:

- Enable the **Content Synchronization** plug-in in Directory Server and optionally create a new user which the client will use to bind to Directory Server. The account must have permissions to read the content in the directory.
- Configure the client. For example, set the search base for a subtree to synchronize. For further details, see your client's documentation.

## 17.1. CONFIGURING THE CONTENT SYNCHRONIZATION PLUG-IN USING THE COMMAND LINE

To configure the **Content Synchronization** plug-in using the command line:

1. The **Content Synchronization** plug-in requires the **Retro Changelog** plug-in to log the **nsuniqueid** attribute:

- a. To verify if the retro changelog is already enabled, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin retro-changelog
show
...
nsslapd-pluginEnabled: off
```

If the **nsslapd-pluginEnabled** parameter is set to **off**, the retro changelog is disabled. To enable, see [Section 15.21.1, “Enabling the Retro Changelog Plug-in”](#).

- b. Add the **nsuniqueid** attribute to retro changelog plug-in configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin retro-changelog
set --attribute nsuniqueid:targetUniqueld
```

- c. Optionally, apply the following recommendations for improved performance:

- i. Set maximum validity for entries in the retro change log. For example, to set 2 days (**2d**):

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 2d
```

- ii. If you know which back end or subtree clients access to synchronize data, limit the scope of the **Retro Changelog** plug-in. For example, to exclude the **cn=demo,dc=example,dc=com** subtree, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin retro-changelog set --exclude-suffix "cn=demo,dc=example,dc=com"
```

2. Enable the **Content Synchronization** plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin set --enabled on "Content Synchronization"
```

3. Using the defaults, Directory Server creates an access control instruction (ACI) in the **oid=1.3.6.1.4.1.4203.1.9.1.1,cn=features,cn=config** entry that enables all users to use the **SyncRepl** protocol:

```
aci: (targetattr != "aci")(version 3.0; acl "Sync Request Control";  
    allow( read, search ) userdn = "ldap://all";)
```

Optionally, update the ACI to limit using the **SyncRepl** control. For further details about ACIs, see [Section 18.10, “Defining Bind Rules”](#).

4. Restart Directory Server:

```
# dsctl instance_name restart
```

Clients are now able to synchronize data with Directory Server using the **SyncRepl** protocol.

# CHAPTER 18. MANAGING ACCESS CONTROL

This chapter describes how you use Access Control Instructions (ACI) in Red Hat Directory Server to manage access to entries.

## 18.1. ACCESS CONTROL PRINCIPLES

When Directory Server receives a request, it uses the authentication information provided by the user in the bind operation and the ACIs defined in the directory to allow or deny access to the requested entry or attribute. The server can allow or deny permissions for actions, such as **read**, **write**, **search**, and **compare**. The permission level granted to a user depends on the authentication information provided.

Access control in Directory Server enables you to set precise rules on when the ACIs are applicable:

- For the entire directory, a subtree, or specific entries
- For a specific user, all users belonging to a specific group or role, or all users in the directory
- For a specific location, such as an IP address, an IP range, or a DNS name.

Note that load balancers can affect location-specific rules.



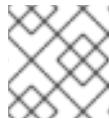
### IMPORTANT

Complex ACIs are difficult to read and understand. Instead of one complex ACI, you can write multiple simple rules to achieve the same effect. However, a higher number of ACIs also increases the costs of ACI processing.

## 18.2. ACI PLACEMENT

Directory Server stores ACIs in the multi-valued **aci** operational attribute in directory entries. To set an ACI, add the **aci** attribute to the corresponding directory entry. Directory Server applies the ACIs:

- Only to the entry that contains the ACI, if it does not have any child entries. For example, if a client requires access to the **uid=user\_name,ou=People,dc=example,dc=com** object, and an ACI is only set on **dc=example,dc=com** and not on any child entries, only this ACI is applied.



### NOTE

ACIs with **add** permissions also apply to child entries created in future.

- To the entry that contains the ACI and to all entries below it, if it has child entries. 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.

For example, ACIs are set on the **dc=example,dc=com** and the **ou=People,dc=example,dc=com** entry: If a client wants to access the **uid=user\_name,ou=People,dc=example,dc=com** object, which has no ACI set, Directory Server first creates a set with the ACIs from **dc=example,dc=com** and **ou=People,dc=example,dc=com**. Directory Server builds the list of applicable ACIs bottom-up from the target entry up to the top suffix. However, consider this list as a set, and the client application should not anticipate any order into ACI evaluation.

The server selects the ACIs that match the resource entry that creates the final set of applicable ACIs from this initial set. Then it first evaluates the ACIs that deny permission. If a **DENY** ACI has been successfully evaluated, the operation fails. If no **DENY** ACI is found, Directory Server checks if an ACI exists that grants **ALLOW** permissions. If at least one of the ACIs allows access, Directory Server grants access. If no ACI grants **ALLOW** permissions, Directory Server refuses access, and the operation fails.



#### NOTE

ACIs set in the **rootDSE** entry apply only to this entry.

An ACI created on an entry can be set not to apply directly to that entry but rather to some or all of the entries in the subtree below. The advantage of this approach is that general ACIs can be placed higher in the directory tree to have effect on entries located lower in the tree. For example, an ACI that targets entries that include the **inetOrgPerson** object class can be created at the level of an **organizationalUnit** entry or a **locality** entry.



#### NOTE

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, place them to leaf entries as closely as possible.

### 18.3. ACI STRUCTURE

The **aci** attribute uses the following syntax:

```
(target_rule) (version 3.0; acl "ACL_name"; permission_rule bind_rules;)
```

- **target\_rule** specifies the entry, attributes, or set of entries and attributes for which to control access. For details, see [Section 18.8, “Defining Targets”](#).
- **version 3.0** is a required string which identifies the ACI version.
- **aci "ACL\_name"** sets a name or string that describes the ACI.
- **permission\_rule** sets what rights, such as **read** or **write**, are allowed or denied. For details, see [Section 18.9, “Defining Permissions”](#).
- **bind\_rules** specifies which rules must match during the bind to allow or deny access. For details, see [Section 18.10, “Defining Bind Rules”](#).



#### NOTE

The permission and the bind rule pair are called an access control rule.

To efficiently set multiple access controls for a given target, you can set multiple access control rules for each target:

```
(target_rule)(version 3.0; acl "ACL_name"; permission_rule bind_rules; permission_rule bind_rules;
... ;)
```

## 18.4. ACI EVALUATION

To evaluate the access rights to a particular entry, the server creates a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored in Directory Server. ACIs are evaluated across all databases for a particular instance but not across different instances.

Directory Server evaluates this list of ACIs 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.

In Directory Server, the **deny** permission in ACIs take precedence over the **allow** permission. For example, if you deny write permission at the directory's root level, none of the users can write to the directory, regardless if an other ACI grants this permission. To grant a specific user write permissions to the directory, you have to add an exception to the original denying rule to allow the user to write in that directory.



### NOTE

For improved ACIs, use fine-grained **allow** rules instead of **deny** rules.

## 18.5. LIMITATIONS OF ACIS

When you set ACIs, the following restrictions apply:

- If your directory database is distributed over multiple servers, the following restrictions apply to the keywords you can use in ACIs:
  - ACIs depending on group entries using the **groupdn** keyword must be located on the same server as the group entry.  
If the group is dynamic, all members of the group must have an entry on the server. Member entries of static groups can be located on the remote server.
  - ACIs depending on role definitions using the **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 match values stored in the target entry with values stored in the entry of the bind user by, for example, using the **userattr** keyword. In this case, access is evaluated normally even if the bind user does not have an entry on the server that stores the ACI.

For further details, see [Section 2.3.3, “Database Links and Access Control Evaluation”](#).

- You cannot use virtual attributes, such as Class of Service (CoS) attributes, in the following ACI keywords:
  - **targetfilter**
  - **targattrfilters**
  - **userattr**

For details, see [Chapter 8, \*Organizing and Grouping Entries\*](#).

- Access control rules are evaluated only on the local server. For example, if you specify the host name of a server in LDAP URLs in ACI keywords, the URL will be ignored.

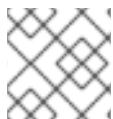
## 18.6. HOW DIRECTORY SERVER HANDLES ACIS IN A REPLICATION TOPOLOGY

ACIs are stored in ***aci*** attributes of entries. Therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated.

ACIs are always evaluated on the server that resolves the incoming LDAP requests. When a consumer server receives an update request, it returns a referral to the supplier server before evaluating whether the request can be serviced on the supplier.

## 18.7. MANAGING ACIS

This section describes how to manage ACIs.



### NOTE

Managing Directory Server ACIs is not supported in the web console.

### 18.7.1. Displaying ACIs

Use the **ldapsearch** utility to display ACI using the command line. For example, to display the ACIs set on **dc=example,dc=com** and sub-entries:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
-b "dc=example,dc=com" -s sub '(aci=*)' aci
```

### 18.7.2. Adding an ACI

Use the **ldapmodify** utility to add an ACI. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword") (version 3.0; acl "Allow users updating their password";
allow (write) userdn= "ldap:///self";)
```

### 18.7.3. Deleting an ACI

To delete an ACI using the command line:

1. Display the ACIs set on the entry. See [Section 18.7.1, “Displaying ACIs”](#).
2. Delete the ACI:
  - o If only one ***aci*** attribute is set on the entry or you want to remove all ACIs from the entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: delete
delete: aci
```

- If multiple ACIs exist on the entry and you want to delete a specific ACI, specify the exact ACI:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
delete: aci
aci: (targetattr="userPassword") (version 3.0; acl "Allow users
updating their password"; allow (write) userdn= "ldap:///self";)
```

For further details about deleting attributes, see [Section 3.4.3, “Deleting Attributes from an Entry”](#).

#### 18.7.4. Updating an ACI

To update an ACI using the command line:

1. Delete the existing ACI. See [Section 18.7.3, “Deleting an ACI”](#).
2. Add a new ACI with the updated settings. See [Section 18.7.2, “Adding an ACI”](#).

### 18.8. DEFINING TARGETS

Target rules in an ACI define to which entries Directory Server applies the ACI. If you do not set a target, the ACI applies to the entry containing the **aci** attribute and to entries below.

In an ACI, the following highlighted part is the target rule:

```
(target_rule)(version 3.0; acl "ACL_name"; permission_rule bind_rules);
```

For complex ACIs, Directory Server supports multiple target rules with different keywords in an ACI:

```
(target_rule_1)(target_rule_2)...)(version 3.0; acl "ACL_name"; permission_rule bind_rules);
```

If you specify multiple target rules, the order is not relevant. Note that you can use each of the following keywords only once in an ACI:

- **target**
- **targetattr**
- **targetattrfilters**
- **targetfilter**
- **target\_from**
- **target\_to**

#### Syntax

The general syntax of a target rule is:

```
(keyword comparison_operator "expression")
```

- **keyword**: Sets the type of the target. See [Section 18.8.1, “Frequently Used Target Keywords”](#).

- **comparison\_operator**: Valid values are `=` and `!=` and indicate whether or not the target is the object specified in the expression.



### WARNING

For security reasons, Red Hat recommends not using the `!=` operator, because it allows the specified operation on all other entries or attributes. For example:

```
(targetattr != "userPassword");(version 3.0; acl "example"); allow (write)
... );
```

The previous example allows users to set, update, or delete any attribute except the **userPassword** attribute under the Distinguished Name (DN) you set the ACI. However, this also enables users, for example, to add an additional **aci** attribute that allows write access to this attribute as well.

- **expression**: Sets the target and must be surrounded by quotation marks. The expression itself depends on the keyword you use.

## 18.8.1. Frequently Used Target Keywords

Administrators frequently use the following target keywords:

- **target**: See [Section 18.8.1.1, “Targeting a Directory Entry”](#).
- **targetattr**: See [Section 18.8.1.2, “Targeting Attributes”](#).
- **targetfilter**: See [Section 18.8.1.3, “Targeting Entries and Attributes Using LDAP Filters”](#).
- **targattrfilters**: See [Section 18.8.1.4, “Targeting Attribute Values Using LDAP Filters”](#).

### 18.8.1.1. Targeting a Directory Entry

To control access based on a DN and the entries below it, use the **target** keyword in the ACI. A target rule which uses the **target** keyword takes a DN as expression:

```
(target comparison_operator "ldap:///distinguished_name")
```



### NOTE

You must set the ACI with the **target** keyword on the DN you are targeting or a higher-level DN of it. For example, if you target `ou=People,dc=example,dc=com`, you must either set the ACI on `ou=People,dc=example,dc=com` or `dc=example,dc=com`.

### Example 18.1. Using the **target** Keyword

To enable users that are stored in the **ou=People,dc=example,dc=com** entry to search and display all attributes in their own entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com") (version 3.0;
acl "Allow users to read and search attributes of own entry"; allow (search, read)
(userdn = "ldap:///self");)
```

## Using Wildcards with the **target** Keyword

You can use the \* wildcard character target multiple entries.

The following target rule example matches all entries in **ou=People,dc=example,dc=com** whose **uid** attribute is set to a value that starts with the letter **a**:

```
(target = "ldap:///uid=a*,ou=People,dc=example,dc=com")
```

Depending on the position of the wildcard, the rule not only applies to attribute values, but also to the full DN. Therefore, you can use the wildcard as a substitute for portions of the DN.

### Example 18.2. Targeting a Directory Entries Using Wildcards

The following rule targets all entries in the **dc=example,dc=com** tree with a matching **uid** attribute and not only entries which are stored in the **dc=example,dc=com** entry itself:

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

The previous target rule matches multiple entries, such as:

- **uid=user\_name,dc=example,dc=com**
- **uid=user\_name,ou=People,dc=example,dc=com**
- **uid=user\_name2,dc=example,dc=com**



## IMPORTANT

Directory Server does not support wildcards in the suffix part of a DN. For example, if your directory's suffix is **dc=example,dc=com**, you cannot use a target with a wildcard in this suffix, such as **(target = "ldap:///dc=\*.com")**.

### 18.8.1.2. Targeting Attributes

To limit access in an ACI to certain attributes, use the **targetattr** keyword. For example, this keyword defines:

- In a read operation, what attributes will be returned to a client
- In a search operation, what attributes will be searched

- In a write operation, what attributes can be written to an object
- In an add operation, what attributes can be added when creating a new object



### NOTE

In certain situations, you can use the **targetattr** keyword to secure ACIs by combining other target keywords with **targetattr**. See [Section 18.8.3, “Advanced Usage of Target Rules”](#) for examples.



### IMPORTANT

In **read** and **search** operations, the default targets no attribute. An ACI without a **targetattr** keyword is only useful for ACIs with rights affecting a complete entry, such as **add** or **delete**.

To separate multiple attributes in a target rule that uses the **targetattr** keyword, use **||**:

```
(targetattr comparison_operator "attribute_1 || attribute_2 || ...")
```

The attributes set in the expression must be defined in the schema.



### NOTE

The attributes specified in the expression apply to the entry on which you create the ACI and to all entries below it if not restricted by further target rules.

#### Example 18.3. Using the **targetattr** Keyword

To enable users stored in **dc=example,dc=com** and all subentries to update the **userPassword** attribute in their own entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "userPassword") (version 3.0;
    acl "Allow users updating own userPassword";
    allow (write) (userdn = "ldap:///self");)
```

#### Using Wildcards with the **targetattr** Keyword

Using the **\*** wildcard character, you can, for example, target all attributes:

```
(targetattr = "*")
```



## WARNING

For security reasons, do not use wildcards with the **targetattr**, because it allows access to all attributes, including operational attributes. For example, if users can add or modify all attributes, users might create additional ACI and increase their own permissions.

### 18.8.1.3. Targeting Entries and Attributes Using LDAP Filters

To target a group of entries that match a certain criteria, use the **targetfilter** keyword with an LDAP filter:

```
(targetfilter comparison_operator "LDAP_filter")
```

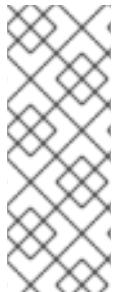
The filter expression is a standard LDAP search filter, as described in [Chapter 14, Finding Directory Entries](#).

#### Example 18.4. Using the **targetfilter** Keyword

To grant permissions to members of the **cn=Human Resources,dc=example,dc.com** group to modify all entries having the **department** attribute set to **Engineering** or **Sales**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetfilter = "(|(department=Engineering)(department=Sales))")
(version 3.0; acl "Allow HR updating engineering and sales entries";
allow (write) (groupdn = "ldap:///cn=Human Resources,dc=example,dc.com");)
```

The **targetfilter** keyword targets whole entries. If you combine it with the **targetattr** keyword, the ACI applies only to a subset of attributes of the targeted entries. See [Section 18.8.3.3, “Targeting Certain Attributes of Entries Matching a Filter”](#).



## NOTE

Using LDAP filters is useful when targeting entries and attributes that are spread across the directory. However, 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, verify that they target the correct entries and attributes by using the same filter, for example, in an **ldapsearch** operation.

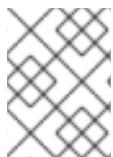
### Using Wildcards with the **targetfilter** Keyword

The **targetfilter** keyword supports wildcards similarly to standard LDAP filters. For example, to target all **uid** attributes whose value starts with **adm**:

```
(targetfilter = "(uid=adm*) ...")
```

#### 18.8.1.4. Targeting Attribute Values Using LDAP Filters

You can use access control to target specific values of attributes. This means that you can grant or deny permissions on an attribute if that attribute's value meets the criteria that is defined in the ACI. An ACI that grants or denies access based on an attribute's value is called a value-based ACI.



#### NOTE

This applies only to **ADD** and **DEL** operations. You cannot limit search rights by specific values.

To create a value-based ACI, use the **targattrfilters** keyword with the following syntax:

- For one operation with one attribute and filter combination:

```
(targattrfilters="operation=attribute:filter")
```

- For one operation with multiple attribute and filter combinations:

```
(targattrfilters="operation=attribute_1:filter_1 && attribute_2:filter_2 ... && attribute_m:filter_m")
```

- For two operations, each with multiple attribute and filter combinations:

```
(targattrfilters="operation_1=attribute_1_1:filter_1_1 && attribute_1_2:filter_1_2 ... && attribute_1_m:filter_1_m , operation_2=attribute_2_1:filter_2_1 && attribute_2_2:filter_2_2 ... & attribute_2_n:filter_2_n")
```

In the previous syntax examples, you can set the operations either to **add** or **del**. The **attribute:filter** combination sets the filter and the attribute the filter is applied to.

The following describes how filter must match:

- When creating an entry and a filter applies to an attribute in the new entry, then each instance of that attribute must match the filter.
- When deleting an entry and a filter applies to an attribute in the entry, then each instance of that attribute must also match the filter.
- When modifying an entry and the operation adds an attribute, then the **add** filter that applies to that attribute must match.
- If the operation deletes an attribute, then the **del** filter that applies to that attribute must match. If the individual values of an attribute already present in the entry are replaced, then both the **add** and **del** filters must match.

#### Example 18.5. Using the targattrfilters Keyword

To create an ACI that enables users to add any role to their own entry, except the **Admin** role, and to add the **telephone** attribute, as long as the value begins with the **123** prefix:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattrfilters="add=nsroledn:(!(nsroledn=cn=Admin)) &&
telephoneNumber:(telephoneNumber=123*)") (version 3.0;
acl "Allow adding roles and telephone";
allow (add) (userdn = "ldap:///self");)
```

## 18.8.2. Further Target Keywords

This section describes target keywords that are less-frequently used.

### 18.8.2.1. Targeting Source and Destination DNs

In certain situations, administrators want to allow users to move directory entries. Using the **target\_from** and **target\_to** keywords in an ACI, you can specify the source and destination of the operation, however, without enabling the user:

- To move entries from a different source as set in the ACI.
- To move entries to a different destination as set in the ACI.
- To delete existing entries from the source DN.
- To add new entries to the destination DN.

#### Example 18.6. Using the **target\_from** and **target\_to** Keywords

For example, to enable the **uid=user,dc=example,dc=com** account to move user accounts from the **cn=staging,dc=example,dc=com** entry to **cn=people,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target_from="ldap:///uid=*,cn=staging,dc=example,dc=com")
(target_to="ldap:///cn=People,dc=example,dc=com")
(version 3.0; acl "MODDN from"; allow (moddn))
userdn="ldap:///uid=user,dc=example,dc=com";)
```



#### NOTE

ACIs apply only to the subtree where they are defined. In the previous example, the ACI is applied only to the **dc=example,dc=com** subtree.

If the **target\_from** or **target\_to** keyword is not set, the ACI matches any source or destination.

## 18.8.3. Advanced Usage of Target Rules

By combining multiple keywords, you can create complex target rules. This section provides examples of the advanced usage of target rules.

### 18.8.3.1. Delegating Permissions to Create and Maintain Groups

In certain situations, administrators want to delegate permissions to other accounts or groups. By combining target keywords, you can create secure ACIs that solve this request.

#### Example 18.7. Delegating Permissions to Create and Maintain Groups

To enable the **uid=user,ou=People,dc=example,dc=com** account to create and update groups in the **ou=groups,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///cn=*,ou=Groups,dc=example,dc=com")
  (targetattr="add=objectclass:(!(objectclas=top)(objectclass=groupOfUniqueNames)))
  (targetattr="cn || uniqueMember || objectClass")
  (version 3.0; acl "example"; allow (read, search, write, add)
  (userdn = "ldap:///uid=test,ou=People,dc=example,dc=com");)
```

For security reasons, the previous example adds certain limitations. The **uid=test,ou=People,dc=example,dc=com** user:

- Can create objects that must contain the **top** and **groupOfUniqueNames** object classes.
- Cannot add additional object classes, such as **account**. For example, this prevents if you use Directory Server accounts for local authentication, to create new users with an invalid user ID, such as **0** for the **root** user.

The **targetfilter** rule ensures that the ACI entry applies only to entries with the **groupofuniqueNames** object class and the **targetattrfilter** rule ensures that no other object class can be added.

### 18.8.3.2. Targeting Both an Entry and Attributes

The **target** controls access based on a DN. However, if you use it in combination with a wildcard and the **targetattr** keyword, you can target both entries and attributes.

#### Example 18.8. Targeting Both an Entry and Attributes

To enable the **uid=user,ou=People,dc=example,dc.com** user to read and search members of groups in all organizational units in the **dc=example,dc=com** subtree:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=*,dc=example,dc=com")(targetattr="member" || "cn") (version 3.0;
  acl "Allow uid=user to search and read members of groups";
  allow (read, search) (userdn = "ldap:///uid=user,ou=People,dc=example,dc.com");)
```

### 18.8.3.3. Targeting Certain Attributes of Entries Matching a Filter

If you combine the **targetattr** and **targetfilter** keywords in two target rules, you can target certain attributes in entries that match a filter.

#### Example 18.9. Targeting Certain Attributes of Entries Matching a Filter

To allow members of the **cn=Engineering Admins,dc=example,dc=com** group to modify the **jpegPhoto** and **manager** attributes of all entries having the **department** attribute set to **Engineering**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "jpegPhoto || manager")
(targetfilter = "(department=Engineering)") (version 3.0;
acl "Allow engineering admins updating jpegPhoto and manager of department members";
allow (write) (groupdn = "ldap:///cn=Engineering Admins,dc=example,dc.com");)
```

### 18.8.3.4. Targeting a Single Directory Entry

To target a single directory entry, combine the **targetattr** and **targetfilter** keywords.

#### Example 18.10. Targeting a Single Directory Entry

To enable the **uid=user,ou=People,dc=example,dc=com** user to read and search the **ou** and **cn** attributes in the **ou=Engineering,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=Engineering,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "ou || cn")
(targetfilter = "(ou=Engineering)") (version 3.0;
acl "Allow uid=user to search and read engineering attributes";
allow (read, search) (userdn = "ldap:///uid=user,ou=People,dc=example,dc.com");)
```

To enable the previous example to target only the **ou=Engineering,dc=example,dc=com** entry, sub-entries in **ou=Engineering,dc=example,dc=com** must not have the **ou** attribute set to **Engineering**.



#### IMPORTANT

These kinds of ACIs can fail if the structure of your directory changes.

Alternatively, you can create a bind rule that matches the user input in the bind request with an attribute value that is stored in the targeted entry. See [Section 18.10.2.1, “Defining Access Based on Value Matching”](#).

## 18.9. DEFINING PERMISSIONS

Permission rules define the rights that are associated with the ACI and whether access is allowed or denied.

In an ACI, the following highlighted part is the permission rule:

```
(target_rule) (version 3.0; acl "ACL_name"; permission_rule bind_rules;)
```

### Syntax

The general syntax of a permission rule is:

```
permission (rights)
```

- **permission:** Sets if the ACI allows or denies permission.
- **rights:** Sets the rights which the ACI allows or denies. See [Section 18.9.1, “User rights”](#).

### Example 18.11. Defining Permissions

To enable users stored in the **ou=People,dc=example,dc=com** entry to search and display all attributes in their own entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com") (version 3.0;
      acl "Allow users to read and search attributes of own entry"; allow (search, read)
      (userdn = "ldap:///self");)
```

### 18.9.1. User rights

The rights in a permission rule define what operations are granted or denied. In an ACI, you can set one or multiple of the following rights:

**Table 18.1. User Rights**

Right	Description
read	Sets whether users can read directory data. This permission applies only to search operations in LDAP.
write	Sets whether users can modify an entry by adding, modifying, or deleting attributes. This permission applies to the <b>modify</b> and <b>modrdn</b> operations in LDAP.
add	Sets whether users can create an entry. This permission applies only to the <b>add</b> operation in LDAP.

Right	Description
delete	Sets whether users can delete an entry. This permission applies only to the <b>delete</b> operation in LDAP.
search	Sets whether users can search for directory data. To view data returned as part of a search result, assign <b>search</b> and <b>read</b> rights. This permission applies only to search operations in LDAP.
compare	Sets whether the users can compare data they supply with data stored in the directory. With <b>compare</b> 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 in LDAP.
selfwrite	Sets whether users can add or delete their own DN from a group. This right is used only for group management.
proxy	Sets whether the specified DN can access the target with the rights of another entry. The <b>proxy</b> right is granted within the scope of the ACL, and the user or group who has the right granted can run commands as any Directory Server user. You cannot restrict the proxy rights to certain users.  For security reasons, set ACIs that use the <b>proxy</b> right at the most targeted level of the directory.
all	Sets all of the rights, except <b>proxy</b> .

### 18.9.2. Rights Required for LDAP Operations

This section describes the rights you must 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 that you want to add.
  - Grant **write** permission on the value of each attribute in the entry. This right is granted by default but can be restricted using the **targattrfilters** keyword.
- Deleting an entry:
  - Grant **delete** permission on the entry that you want to delete.
  - Grant **write** permission on the value of each attribute in the entry. This right is granted by default but can 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 can 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 that is used in the new RDN.
- Grant **write** permission on the attribute type that is 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 that is used in the new RDN. This right is granted by default but can 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.

### 18.9.3. Access Control and the modrdn Operation

To explicitly deny **modrdn** operations using ACIs, target the relevant entries but omit the **targetattr** keyword. For example, to add an ACI that defines the **cn=example,ou=Groups,dc=example,dc=com** group, cannot rename entries in **ou=people,dc=example,dc=com** which contain the **cn** attribute:

```
ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=*,ou=people,dc=example,dc=com")
  (version 3.0; acl "Deny modrdn rights to the example group";
  deny(write) groupdn="ldap:///cn=example,ou=groups,dc=example,dc=com";)
```

## 18.10. DEFINING BIND RULES

The bind rules in an ACI define the required bind parameters that must meet so that Directory Server applies the ACI. For example, you can set bind rules based on:

- DNs
- Group memberships or assigned roles
- Locations from which an entry must bind
- Types of authentication that must be in use during the bind
- Times or days on which the bind occurs

In an ACI, the following highlighted part is the bind rule:

```
(target_rule) (version 3.0; acl "ACL_name"; permission_rule bind_rules;
```

### Syntax

The general syntax of a bind rule is:

`keyword comparison_operator "expression"`

- **keyword:** Sets the type of the bind operation. See [Section 18.10.1, “Frequently Used Bind Rules”](#).
- **comparison\_operator:** Valid values are `=` and `!=` and indicate whether or not the target is the object specified in the expression. If a keyword supports additional comparison operators, it is mentioned in the corresponding section.
- **expression:** Sets the expression and must be surrounded by quotation marks. The expression itself depends on the keyword you use.

## 18.10.1. Frequently Used Bind Rules

Administrators frequently use the following bind keywords:

- **userdn:** See [Section 18.10.1.1, “Defining User-based Access”](#).
- **groupdn:** See [Section 18.10.1.2, “Defining Group-based Access”](#).

Additionally, bind rules are frequently combined using Boolean operators. For details, see [Section 18.10.3, “Combining Bind Rules Using Boolean Operators”](#).

### 18.10.1.1. Defining User-based Access

The **userdn** keyword enables you to grant or deny access based on one or multiple DNs and uses the following syntax:

`userdn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."`

Set the DN in the expression to:

- A DN: See [Section 18.10.1.1.1, “Using a DN with the \*\*userdn\*\* Keyword”](#).
- An LDAP filter: See [Section 18.10.1.1.2, “Using the \*\*userdn\*\* Keyword with an LDAP Filter”](#).
- The **anyone** alias: See [Section 18.10.1.1.3, “Granting Anonymous Access”](#).
- The **all** alias: See [Section 18.10.1.1.4, “Granting Access to Authenticated Users”](#).
- The **self** alias: See [Section 18.10.1.1.5, “Enabling Users to Access Their Own Entries”](#).
- The **parent** alias: See [Section 18.10.1.1.6, “Setting Access for Child Entries of a User”](#).



#### NOTE

Do not specify a host name or port number within the LDAP URL. The URL always applies to the local server.

#### 18.10.1.1.1. Using a DN with the **userdn** Keyword

Set the **userdn** keyword to a DN to apply the ACI only to the matching entry. To match multiple entries, use the `*` wildcard in the DN.

Using the **userdn** keyword with a DN must match the following syntax:

`userdn comparison_operator ldap:///distinguished_name`

#### Example 18.12. Using a DN with the `userdn` Keyword

To enable the `uid=admin,ou=People,dc=example,dc=com` user to read the `manager` attribute of all other users in the `ou=People,dc=example,dc=com` entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0; acl "Allow uid=admin reading manager attribute";
allow (search, read) userdn = "ldap:///uid=admin,ou=People,dc=example,dc=com";)
```

#### 18.10.1.1.2. Using the `userdn` Keyword with an LDAP Filter

If you want to dynamically allow or deny permissions to users, use the `userdn` keyword with an LDAP filter:

`userdn comparison_operator "ldap:///distinguished_name??scope?(filter)"`



#### NOTE

The LDAP filter supports the \* wildcard.

#### Example 18.13. Using the `userdn` Keyword with an LDAP Filter

To enable users who have the `department` attribute set to **Human Resources** to update the `homePostalAddress` attribute of users in the `ou=People,dc=example,dc=com` entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="homePostalAddress") (version 3.0;
acl "Allow HR setting homePostalAddress"; allow (write)
userdn = "ldap:///ou=People,dc=example,dc=com??sub?(department=Human Resources);")
```

#### 18.10.1.1.3. Granting Anonymous Access

In certain situations, administrators want to configure anonymous access to data in the directory. Anonymous access means that it is possible to bind to the directory by providing:

- No bind DN and password
- A valid bind DN and password

To configure anonymous access, use the `ldap://anyone` expression with the `userdn` keyword in a bind rule:

```
userdn comparison_operator "ldap:///anyone"
```

#### Example 18.14. Granting Anonymous Access

To enable anyone without authentication to read and search the ***sn***, ***givenName***, and ***telephoneNumber*** attributes in the **ou=People,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="sn" || targetattr="givenName" || targetattr = "telephoneNumber")
(version 3.0; acl "Anonymous read, search for names and phone numbers";
allow (read, search) userdn = "ldap:///anyone")
```

#### 18.10.1.1.4. Granting Access to Authenticated Users

In certain situations, administrators want to grant permission to any user who is able to successfully bind to Directory Server, except anonymous binds. To configure this feature, use the **ldap://all** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap://all"
```

#### Example 18.15. Granting Access to Authenticated Users

To enable authenticated users to add and remove themselves as a member to or from the **ou=example,ou=groups,dc=example,dc=com** group:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=example,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="member") (version 3.0;
acl "Allow users to add/remove themselves from example group";
allow (selfwrite) userdn = "ldap://all")
```

#### 18.10.1.1.5. Enabling Users to Access Their Own Entries

To set ACIs which allow or deny access to users to their own entry, use the **ldap://self** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap://self"
```

#### Example 18.16. Enabling Users to Access Their Own Entries

To enable users in the **ou=People,dc=example,dc=com** entry to update their own ***userPassword*** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
```

```

changetype: modify
add: aci
aci: (targetattr="userPassword") (version 3.0;
acl "Allow users updating their password";
allow (write) userdn = "ldap:///self")

```

#### 18.10.1.6. Setting Access for Child Entries of a User

To specify that users are granted or denied access to an entry only if their bind DN is the parent of the targeted entry, use the **self://parent** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap:///parent"
```

#### Example 18.17. Setting Access for Child Entries of a User

To enable the **cn=user,ou=People,dc=example,dc=com** user to update the **manager** attribute of its own sub-entries, such as **cn=example,cn=user,ou=People,dc=example,dc=com**:

```

# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=user,ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0;
acl "Allow cn=user to update manager attributes";
allow (write) userdn = "ldap:///parent")

```

#### 18.10.1.2. Defining Group-based Access

Group-based ACIs enable you to manage access by adding or removing users to or from a group. To configure an ACI that is based on a group membership, use the **groupdn** keyword. If the user is a member of one or multiple of the specified groups, the ACI matches.

When using the **groupdn** keyword, Directory Server verifies the group membership based on the following attributes:

- ***member***
- ***uniqueMember***
- ***memberURL***
- ***memberCertificateDescription***

Bind rules with the **groupdn** keyword use the following syntax:

```
groupdn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."
```

Set the DN in the expression to:

- A DN. See [Section 18.10.1.2.1, “Using a DN with the \*\*groupdn\*\* Keyword”](#).
- An LDAP filter. See [Section 18.10.1.2.2, “Using the \*\*groupdn\*\* Keyword with an LDAP Filter”](#).

If you set multiple DNs in one bind rule, Directory Server applies the ACI if the authenticated user is a member of one of these groups. To set the user as a member of multiple groups, use multiple **groupdn** keywords and combine them using the Boolean **and** operator. For details, see [Section 18.10.3, "Combining Bind Rules Using Boolean Operators"](#).



#### NOTE

Do not specify a host name or port number within the LDAP URL. The URL always applies to the local server.

##### 18.10.1.2.1. Using a DN with the **groupdn** Keyword

To apply an ACI to members of a group, set the **groupdn** keyword to the group's DN.

The **groupdn** keyword set to a DN uses the following syntax:

```
groupdn comparison_operator ldap:///distinguished_name
```

#### Example 18.18. Using a DN with the **groupdn** Keyword

To enable members of the **cn=example,ou=Groups,dc=example,dc=com** group to search and read the **manager** attribute of entries in **ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0;
acl "Allow example group to read manager attribute";
allow (search, read) groupdn = "ldap:///cn=example,ou=Groups,dc=example,dc=com";)
```

##### 18.10.1.2.2. Using the **groupdn** Keyword with an LDAP Filter

Using an LDAP filter with the **groupdn** keyword, you can define that the authenticated user must be a member of at least one of the groups that the filter search returns, to match the ACI.

The **groupdn** keyword with an LDAP filter uses the following syntax:

```
groupdn comparison_operator "ldap:///distinguished_name??scope?(filter)"
```



#### NOTE

The LDAP filter supports the \* wildcard.

#### Example 18.19. Using the **groupdn** Keyword with an LDAP Filter

To enable members of groups in **dc=example,dc=com** and subtrees, which have the **manager** attribute set to **example**, update the **homePostalAddress** of entries in **ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

```

dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="homePostalAddress") (version 3.0;
    acl "Allow manager=example setting homePostalAddress"; allow (write)
    userdn = "ldap:///dc=example,dc=com??sub?(manager=example)";)

```

## 18.10.2. Further Bind Rules

This section describes bind rules that are less-frequently used.

### 18.10.2.1. Defining Access Based on Value Matching

Use the **userattr** keyword in a bind rule to specify which attribute must match between the entry used to bind to the directory and the targeted entry.

The **userattr** keyword uses the following syntax:

```
userattr comparison_operator "attribute_name#bind_type_or_attribute_value"
```

For further details, see:

- [Section 18.10.2.1.1, “Using the \*\*USERDN\*\* Bind Type”](#)
- [Section 18.10.2.1.2, “Using the \*\*GROUPDN\*\* Bind Type”](#)
- [Section 18.10.2.1.3, “Using the \*\*ROLEDN\*\* Bind Type”](#)
- [Section 18.10.2.1.4, “Using the \*\*SELFDN\*\* Bind Type”](#)
- [Section 18.10.2.1.5, “Using the \*\*LDAPURL\*\* Bind Type”](#)



#### IMPORTANT

By default, Directory Server evaluates access rights on the entry they are created. However, to prevent user objects on the same level, Directory Server does not grant **add** permissions to the entry where you set the ACI, when using the **userattr** keyword. To configure this behavior, use the **userattr** keyword in conjunction with the **parent** keyword and grant the permission additionally on level **0**.

For details about inheritance, see [Section 18.10.2.1.6, “Using the \*\*userattr\*\* Keyword with Inheritance”](#).

#### 18.10.2.1.1. Using the **USERDN** Bind Type

To apply an ACI when the binding user DN matches the DN stored in an attribute, use the **USERDN** bind type.

The **userattr** keyword with the **USERDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#USERDN"
```

### Example 18.20. Using the **USERDN** Bind Type

To grant a manager all permissions to the **telephoneNumber** attribute of its own associates:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "telephoneNumber")
(version 3.0; acl "Manager: telephoneNumber";
allow (all) userattr = "manager#USERDN";)
```

The previous ACI is evaluated to be true if the DN of the user who performs the operation on an entry in **ou=People,dc=example,dc=com**, matches the DN stored in the **manager** attribute of this entry.

#### 18.10.2.1.2. Using the **GROUPDN** Bind Type

To apply an ACI when the binding user DN is a member of a group set in an attribute, use the **GROUPDN** bind type.

The **userattr** keyword with the **GROUPDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#GROUPDN"
```

### Example 18.21. Using the **GROUPDN** Bind Type

To grant users the permission to delete a group entry which they own under the **ou=Social Committee,ou=Groups,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=Social Committee,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ou=Social Committee,ou=Groups,dc=example,dc=com")
(targattrfilters="del=objectClass:(objectClass=groupOfNames)")
(version 3.0; acl "Delete Group";
allow (delete) userattr = "owner#GROUPDN";)
```

The previous ACI is evaluated to be true if the DN of the user who performs the operation is a member of the group specified in the **owner** attribute.

The specified group 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, use the following expression for better performance:

```
userattr comparison_operator "ldap:///distinguished_name?attribute_name#GROUPDN"
```

#### 18.10.2.1.3. Using the **ROLEDN** Bind Type

To apply an ACI when the binding user belongs to a role specified in an attribute, use the **ROLEDN** bind type.

The **userattr** keyword with the **ROLEDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#ROLEDN"
```

#### Example 18.22. Using the ROLEDN Bind Type

To enable users with the **cn=Administrators,dc=example,dc=com** role to search and read the **manager** attribute of entries in **ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (version 3.0; acl "Allow example role owners to read manager attribute";
allow (search, read) userattr = manager#ROLEDN);
```

The specified role can be under any suffix in the database. If you are also 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, use the following expression for better performance:

#### 18.10.2.1.4. Using the SELFDN Bind Type

The **SELFDN** bind type enables you to grant permissions, when the bound user's DN is set in a single-value attribute of the entry.

The **userattr** keyword with the **SELFDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#SELFDN"
```

#### Example 18.23. Using the SELFDN Bind Type

To enable a user to add **ipatokenuniqueid=\*,cn=otp,dc=example,dc=com** entries that have the bind user's DN set in the **ipatokenOwner** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=otp,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ipatokenuniqueid=*,cn=otp,dc=example,dc=com")
(targetfilter = "(objectClass=ipaToken)")(version 3.0;
acl "token-add-delete"; allow (add) userattr = "ipatokenOwner#SELFDN";)
```

#### 18.10.2.1.5. Using the LDAPURL Bind Type

To apply an ACL when the bind DN matches the filter specified in an attribute of the targeted entry, use the **LDAPURL** bind type.

The **userattr** keyword with the **LDAPURL** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#LDAPURL"
```

#### Example 18.24. Using the **LDAPURL** Bind Type

To grant read and search permissions to user objects which contain the **aciurl** attribute set to **ldap://ou=People,dc=example,dc=com??one?(uid=user\*)**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")
(version 3.0; acl "Allow read,search "; allow (read,search)
(userattr = "aciurl#LDAPURL);)
```

#### 18.10.2.1.6. 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 certain situations, administrators 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 using the **userattr** keyword with the **parent** keyword, the syntax is as follows:

```
userattr comparison_operator
"parent[inheritance_level].attribute_name#bind_type_or_attribute_value"
```

- *inheritance\_level*: Comma-separated list that indicates how many levels below the target inherit the ACI. You can include five levels (**0, 1, 2, 3, 4**) below the targeted entry. Zero (**0**) indicates the targeted entry.
- *attribute\_name*: The attribute targeted by the **userattr** or **groupattr** keyword.
- *bind\_type\_or\_attribute\_value*: Sets the attribute value or a bind type, such as **USERDN**.

For example:

```
userattr = "parent[0,1].manager#USERDN"
```

This bind rule is evaluated to be true if the bind DN 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 18.25. Using the **userattr** Keyword with Inheritance

To enable a user to read and search the **cn=Profiles,dc=example,dc=com** entry where the user's DN is set in the **owner** attribute, as well as the first level of child entries which includes **cn=mail,cn=Profiles,dc=example,dc=com** and **cn=news,cn=Profiles,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=Profiles,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*") (version 3.0; acl "Profile access",
allow (read,search) userattr="parent[0,1].owner#USERDN" ;)
```

### 18.10.2.2. Defining Access from Specific IP Addresses or Ranges

The **ip** keyword in a bind rule enables you to grant or deny access from a specific IP address or a range of IP addresses.

Bind rules with the **ip** keyword use the following syntax:

```
ip comparison_operator "IP_address_or_range"
```

#### Example 18.26. Using IPv4 Address Ranges in Bind Rules

To deny access from the **192.0.2.0/24** network to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny 192.0.2.0/24"; deny (all)
(userdn = "ldap:///anyone") and (ip != "192.0.2."));
```

#### Example 18.27. Using IPv6 Address Ranges in Bind Rules

To deny access from the **2001:db8::/64** network to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny 2001:db8::/64"; deny (all)
(userdn = "ldap:///anyone") and (ip != "2001:db8::"));
```

### 18.10.2.3. Defining Access from a Specific Host or Domain

The **dns** keyword in a bind rule enables you to grant or deny access from a specific host or domain.



## WARNING

If Directory Server cannot resolve a connecting IP address to its Fully Qualified Domain Name (FQDN) using DNS, the server does not apply ACLs with the **dns** bind rule for this client.

If client IP addresses are not resolvable using DNS, use the **ip** keyword and IP addresses instead. See [Section 18.10.2.2, “Defining Access from Specific IP Addresses or Ranges”](#).

Bind rules with the **dns** keyword use the following syntax:

```
dns comparison_operator "host_name_or_domain_name"
```

### Example 18.28. Defining Access from a Specific Host

To deny access from the **client.example.com** host to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "") (version 3.0;acl "Deny client.example.com"; deny (all)
(userdn = "ldap:///anyone") and (dns != "client.example.com");)
```

### Example 18.29. Defining Access from a Specific Domain

To deny access from all hosts within the **example.com** domain to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "") (version 3.0;acl "Deny example.com"; deny (all)
(userdn = "ldap:///anyone") and (dns != "*.example.com");)
```

## 18.10.2.4. Requiring a Certain Level of Security in Connections

The security of a connection is determined by its Security Strength Factor (SSF), which sets the minimum key strength required to process operations. Using the **ssf** keyword in a bind rule, you can set that a connection must use a certain level of security. This enables you to force operations, for example password changes, to be performed over an encrypted connection.

The value for the SSF for any operation is the higher of the values between a TLS connection and a SASL bind. This means that if a server is configured to run over TLS and a replication agreement is configured for SASL/GSSAPI, the SSF for the operation is whichever available encryption type is more

secure.

Bind rules with the **ssf** keyword use the following syntax:

**ssf comparison\_operator key\_strength**

You can use the following comparison operators:

- **=** (equal to)
- **!** (not equal to)
- **<** (less than)
- **>** (greater than)
- **<=** (less than or equal to)
- **>=** (greater than or equal to)

If the **key\_strength** parameter is set to **0**, no secure operation is required for the LDAP operation.

#### Example 18.30. Requiring a Certain Level of Security in Connections

To configure that users in the **dc=example,dc=com** entry can only update their **userPassword** attribute when the SSF is **128** or higher:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "userPassword") (version 3.0;
    acl "Allow users updating own userPassword";
    allow (write) (userdn = "ldap:///self") and (ssf >= "128");)
```

#### 18.10.2.5. Defining Access at a Specific Day of the Week

The **dayofweek** keyword in a bind rule enables you to grant or deny access based on the day of the week.



#### NOTE

Directory Server uses the time on the server to evaluate the ACI; not the time on the client.

Bind rules with the **dayofweek** keyword use the following syntax:

**dayofweek comparison\_operator "comma-separated\_list\_of\_days"**

#### Example 18.31. Granting Access on Specific Days of the Week

To deny access for the **uid=user,ou=People,dc=example,dc=com** user entry to bind to the server on Saturdays and Sundays:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (version 3.0; acl "Deny access on Saturdays and Sundays";
deny (all)
(userdn = "ldap:///uid=user,ou=People,dc=example,dc=com") and
(dayofweek = "Sun,Sat");)
```

#### 18.10.2.6. Defining Access at a Specific Time of Day

The **timeofday** keyword in a bind rule enables you to grant or deny access based on the time of day.



#### NOTE

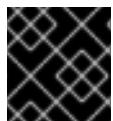
Directory Server uses the time on the server to evaluate the ACL; not the time on the client.

Bind rules with the **timeofday** keyword use the following syntax:

```
timeofday comparison_operator "time"
```

You can use the following comparison operators:

- **=** (equal to)
- **!** (not equal to)
- **<** (less than)
- **>** (greater than)
- **<=** (less than or equal to)
- **>=** (greater than or equal to)



#### IMPORTANT

The **timeofday** keyword requires that you specify the time in 24-hour format.

#### Example 18.32. Defining Access at a Specific Time of a Day

To deny access for the **uid=user,ou=People,dc=example,dc=com** user entry to bind to the server between 6pm and 0am:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
```

```
aci: (version 3.0; acl "Deny access between 6pm and 0am";
deny (all)
(userdn = "ldap:///uid=user,ou=People,dc=example,dc=com") and
(timeofday >= "1800" and timeofday < "2400");)
```

### 18.10.2.7. Defining Access Based on the Authentication Method

The **authmethod** keyword in a bind rule sets what authentication method a client must use when connecting to the server, to apply the ACI.

Bind rules with the **authmethod** keyword use the following syntax:

```
authmethod comparison_operator "authentication_method"
```

You can set the following authentication methods:

- **none**: Authentication is not required and represents anonymous access. This is the default.
- **simple**: The client must provide a user name and password to bind to the directory.
- **SSL**: The client must bind to the directory using a TLS certificate either in a database, smart card, or other device. For details about certificate-based authentication, see [Section 9.9, “Using Certificate-based Client Authentication”](#).
- **SASL**: The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. When you use this authentication method in a bind rule, additionally specify the SASL mechanism, such as **EXTERNAL**.

#### Example 18.33. Enabling Access Only for Connections Using the **EXTERNAL** SASL Authentication Method

To deny access to the server if the connection does not use a certificate-based authentication method or SASL:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (version 3.0; acl "Deny all access without certificate"; deny (all)
(authmethod = "none" or authmethod = "simple");)
```

### 18.10.2.8. Defining Access Based on Roles

The **roledn** keyword in a bind rule enables you to grant or deny access to users having one or multiple role sets.



#### NOTE

Red Hat recommends using groups instead of roles. For further details about roles and limitations, see [Section 8.2.1, “About Roles”](#).

Bind rules with the **roledn** keyword use the following syntax:

```
roledn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."
```



#### NOTE

If a DN contains a comma, escape the comma with a backslash.

#### Example 18.34. Defining Access Based on Roles

To enable users that have the **cn=Human Resources,ou=People,dc=example,dc=com** role set in the **nsRole** attribute to search and read the **manager** attribute of entries in **ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0;
acl "Allow manager role to update manager attribute";
allow (search, read) roledn = "ldap:///cn=Human Resources,ou=People,dc=example,dc=com";)
```

### 18.10.3. Combining Bind Rules Using Boolean Operators

When creating complex bind rules, the **AND**, **OR**, and **NOT** Boolean operators enable you to combine multiple keywords.

Bind rules combined with Boolean operators have the following syntax:

```
bind_rule_1 boolean_operator bind_rule_2...
```

#### Example 18.35. Combining Bind Rules Using Boolean Operators

To configure that users which are member of both the **cn=Administrators,ou=Groups,dc=example,com** and **cn=Operators,ou=Groups,dc=example,com** group can read, search, add, update, and delete entries in **ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///ou=People,dc=example,dc=com") (version 3.0;
acl "Allow members of administrators and operators group to manage users";
allow (read, search, add, write, delete)
groupdn = "ldap:///cn=Administrators,ou=Groups,dc=example,com" AND
groupdn = "ldap:///cn=Operators,ou=Groups,dc=example,com";)
```

### How Directory Server Evaluates Boolean Operators

Directory Server evaluates Boolean operators by using the following rules:

- All expressions from left to right.

In the following example, ***bind\_rule\_1*** is evaluated first:

(bind\_rule\_1) OR (bind\_rule\_2)

- From innermost to outermost parenthetical expressions first.

In the following example, ***bind\_rule\_2*** is evaluated first and ***bind\_rule\_3*** second:

(bind\_rule\_1) OR ((bind\_rule\_2) AND (bind\_rule\_3))

- **NOT** before **AND** or **OR** operators.

In the following example, ***bind\_rule\_2*** is evaluated first:

(bind\_rule\_1) AND NOT (bind\_rule\_2)

The **AND** and **OR** operators have no order of precedence.

## 18.11. CHECKING ACCESS RIGHTS ON ENTRIES (GET EFFECTIVE RIGHTS)

Finding the access rights that a user has on attributes within a specific entry offers a convenient way for administrators to find and control the access rights.

*Get effective rights* is a way to extend directory searches to display what access rights – such as read, search, write and self-write, add, and delete – a user has to a specified entry.

In Directory Server, regular users can check their rights over entries which they can view and can check other people's access to their personal entries. The Directory Manager can check rights that one user has over another user.

There are two common situations where checking the effective rights on an entry are useful:

- An administrator can use the get effective rights command in order to better organize access control instructions for the directory. It is frequently necessary to restrict what one group of users can view or edit versus another group. 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. Checking the effective rights for a user or group is one way to verify that the appropriate access controls are in place.
- 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** attributes.

There are three entities involved in a **getEffectiveRights** search. The first is the *requester*, which is the authenticated entry when the **getEffectiveRights** search operation is issued. The second is the *subject* whose rights will be evaluated, it is defined as *authorization DN* in the *GER* control. The third is the *target*, which is defined by the search base, search filter, and attribute list of the request.

### 18.11.1. Rights Shown with a Get Effective Rights Search

Any get effective rights search, when searching for it in the command line, shows the rights that the requestor has to a target entry.

There are two kinds of access rights that can be allowed to any entry. The first are upper-level rights, *rights on the entry itself*, which means that kinds of operations that the User A can perform on User B's entry as a whole. The second level of access rights are more granular, show what *rights for a given attribute* User A has. In this case, User A may have different kinds of access permissions for different attributes in the same entry. Whatever access controls are allowed for a user are the *effective rights* over that entry.

For example:

```
entryLevelRights: vadn
attributeLevelRights: givenName:rscWO, sn:rscW, objectClass:rsc, uid:rsc, cn:rscW
```

[Table 18.2, "Entry Rights"](#) and [Table 18.3, "Attribute Rights"](#) show the access rights to entries and attributes, respectively, that are returned by a get effective rights search.

**Table 18.2. Entry Rights**

Permission	Description
a	Add an entry.
d	Delete this entry.
n	Rename the DN.
v	View the entry.

**Table 18.3. Attribute Rights**

Permission	Description
r	Read.
s	Search.
w	Write ( <b>mod-add</b> ).
o	Obliterate( <b>mod-del</b> ). Analogous to delete.
c	Compare.
W	Self-write.
O	Self-delete.

## 18.11.2. The Format of a Get Effective Rights Search

Get effective rights (sometimes called GER) is an extended directory search; the GER parameters are defined with the **-E** option to pass an LDAP control with the **Idapsearch** command. (If an **Idapsearch** is run without the **-E** option, then, naturally, the entry is returned as normal, without any get effective rights information.)

```
# ldapsearch -x -D bind_dn -W -p server_port -h server_hostname -E
[!]1.3.6.1.4.1.42.2.27.9.5.2=:GER_subject (searchFilter) attributeList
```

- **-b** is the base DN of the subtree or entry used to search for the GER subject.

If the search base is a specific entry DN or if only one entry is returned, then the results show the rights the requester has over that specific entry. If multiple entries beneath the search base match the filter, then the search returns every matching entry, with the rights for the requester over each entry.

- **1.3.6.1.4.1.42.2.27.9.5.2** is the OID for the get effective rights control.
- The exclamation point (!) specifies whether the search operation should return an error if the server does not support this control (!) or if it should be ignored and let the search return as normal (nothing).
- The *GER\_subject* is the person whose rights are being checked. If the *GER\_subject* is left blank (**dn:**), than the rights of an anonymous user are returned.
- An optional *attributeList* limits the get effective rights results to the specified attribute or object class. As with a regular **Idapsearch**, this can give specific attributes, like **mail**. If no attributes are listed, then every present attribute for the entry is returned. Using an asterisk (\*) returns the rights for every possible attribute for the entry, both existing attribute and non-existent attributes. Using an plus sign (+) returns operational attributes for the entry. Examples for checking rights for specific attributes are given in [Section 18.11.3.2, “Examples of Get Effective Rights Searches for Non-Existent Attributes”](#) and [Section 18.11.3.3, “Examples of Get Effective Rights Searches for Specific Attributes or Object Classes”](#).

The crux of a get effective rights search is the ability to check what rights the GER subject (**-E**) has to the targets of the search (**-b**). The get effective rights search is a regular **Idapsearch**, in that it simply looks for entries that match the search parameters and returns their information. The get effective rights option adds extra information to those search results, showing what rights a specific user has over those results. That GER subject user can be the requester himself (**-D** is the same as **-E**) or someone else.

If the requester is a regular user (not the Directory Manager), then the requester can only see the effective that a GER subject has on the requester's own entry. That is, if John Smith runs a request to see what effective rights Babs Jensen has, then he can only get the effective rights that Babs Jensen has on his own entry. All of the other entries return an insufficient access error for the effective rights.

There are three general scenarios for a regular user when running a get effective rights search:

- User A checks the rights that he has over other directory entries.
- User A checks the rights that he has to his personal entry.
- User A checks the rights that User B has to User A's entry.

The get effective rights search has a number of flexible different ways that it can check rights on attributes.

### 18.11.3. Examples of GER Searches

There are a number of different ways to run GER searches, depending on the exact type of information that needs to be returned and the types of entries and attributes being searched.

#### 18.11.3.1. General Examples on Checking Access Rights

One common scenario for effective rights searches is for a regular user to determine what changes he can make to his personal entry.

For example, Ted Morris wants to check the rights he has to his entry. Both the **-D** and **-E** options give his entry as the requester. Since he is checking his personal entry, the **-b** option also contains his DN.

##### Example 18.36. Checking Personal Rights (User A to User A)

```
# ldapsearch -x -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -W -b "uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=People,dc=example,dc=com
givenName: Ted
sn: Morris
ou: IT
ou: People
l: Santa Clara
manager: uid=jsmith,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
```

Ted Morris may, for example, be a manager or work in a department where he has to edit other user's entries, such as IT or human resources. In this case, he may want to check what rights he has to another user's entry, as in [Example 18.37, "Personally Checking the Rights of One User over Another \(User A to User B\)"](#), where Ted (**-D**) checks his rights (**-E**) to Dave Miller's entry (**-b**):

##### Example 18.37. Personally Checking the Rights of One User over Another (User A to User B)

```
# ldapsearch -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -W -b "uid=dmiller,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=dmiller,ou=People,dc=example,dc=com
...
entryLevelRights: vad
```

```
attributeLevelRights: givenName:rscwo, sn:rscwo, ou:rscwo, l:rscwo, manager:rsc,
roomNumber:rscwo, mail:rscwo, facsimileTelephoneNumber:rscwo, objectClass:rscwo, uid:rscwo,
cn:rscwo, userPassword:rswo
```

For all attributes, Ted Morris has read, search, compare, modify, and delete permissions to Dave Miller'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.

The Directory Manager has the ability to check the rights that one user has over another user's entry. In [Example 18.38, "The Directory Manager's Checking the Rights of One User over Another \(User A to User B\)"](#), the Directory Manager is checking the rights that a manager, Jane Smith (-E), has over her subordinate, Ted Morris (-b):

#### **Example 18.38. The Directory Manager's Checking the Rights of One User over Another (User A to User B)**

```
# ldapsearch -p 389 -h server.example.com -D "cn=Directory Manager" -W -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=jsmith,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=People,dc=example,dc=com
...
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:rswo
```

Only an administrator can retrieve the effective rights that a different user has on an entry. If Ted Morris tried to determine Dave Miller's rights to Dave Miller's entry, then he would receive an insufficient access error:

```
# ldapsearch -p 389 -h server.example.com -D "uid=dmiller,ou=people,dc=example,dc=com" -W -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

ldap_search: Insufficient access
ldap_search: additional info: get-effective-rights: requester has no g permission on the entry
```

However, a regular user can run a get effective rights search to see what rights another user has to his personal entry. In [Example 18.39, "Checking the Rights Someone Else Has to a Personal Entry"](#), Ted Morris checks what rights Dave Miller has on Ted Morris's entry.

#### **Example 18.39. Checking the Rights Someone Else Has to a Personal Entry**

```
# ldapsearch -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -W -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=dmiller,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=people,dc=example,dc=com
...
```

```

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

```

In this case, Dave Miller has the right to view the DN of the entry and to read, search, and compare the *ou*, *givenName*, *I*, and other attributes, and no rights to the *userPassword* attribute.

### 18.11.3.2. Examples of Get Effective Rights Searches for Non-Existent Attributes

By default, 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.

Using an asterisk (\*) with the get effective rights search returns every attribute available for the entry, including attributes not set on the entry.

#### Example 18.40. Returning Effective Rights for Non-Existent Attributes

```

# ldapsearch -D "cn=Directory Manager" -W -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "*"

dn: uid=scarter,ou=People,dc=example,dc=com
givenName: Sam
telephoneNumber: +1 408 555 4798
sn: Carter
ou: Accounting
ou: People
l: Sunnyvale
manager: uid=dmiller,ou=People,dc=example,dc=com
roomNumber: 4612
mail: scarter@example.com
facsimileTelephoneNumber: +1 408 555 9700
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: scarter
cn: Sam Carter
userPassword: {SSHA}Xd9Jt8g1UsHC8enNDRemxj3iJPQLItDYdD9A==
entryLevelRights: vadn
attributeLevelRights: objectClass:rscwo, aci:rscwo, sn:rscwo, cn:rscwo, description:rscwo,
seeAlso:rscwo, telephoneNumber:rscwo, userPassword:rscwo, destinationIndicator:rscwo,
facsimileTelephoneNumber:rscwo, internationalISDNNumber:rscwo, l:rscwo, ou:rscwo,
physicalDeliveryOfficeName:rscwo, postOfficeBox:rscwo, postalAddress:rscwo,
postalCode:rscwo, preferredDeliveryMethod:rscwo, registeredAddress:rscwo, st:rscwo,
street:rscwo, teletexTerminalIdentifier:rscwo, telexNumber:rscwo, title:rscwo, x121Address:rscwo,
audio:rscwo, businessCategory:rscwo, carLicense:rscwo, departmentNumber:rscwo,
displayName:rscwo, employeeType:rscwo, employeeNumber:rscwo, givenName:rscwo,
homePhone:rscwo, homePostalAddress:rscwo, initials:rscwo, jpegPhoto:rscwo, labeledURI:rscwo,
manager:rscwo, mobile:rscwo, pager:rscwo, photo:rscwo, preferredLanguage:rscwo, mail:rscwo,
o:rscwo, roomNumber:rscwo, secretary:rscwo, uid:rscwo,x500UniqueIdentifier:rscwo,
userCertificate:rscwo, userSMIMECertificate:rscwo, userPKCS12:rscwo

```

All of the attributes available for the entry, such as **secretary**, are listed, even though that attribute is non-existent.

### 18.11.3.3. Examples of Get Effective Rights Searches for Specific Attributes or Object Classes

Taking the attribute-related GER searches further, it is possible to search for the rights to a specific attribute and set of attributes and to list all of the attributes available for one of the object classes set on the entry.

One of the options listed in the formatting example in [Section 18.11.2, "The Format of a Get Effective Rights Search"](#) is *attributeList*. To return the effective rights for only specific attributes, list the attributes, separated by spaces, at the end of the search command.

#### Example 18.41. Get Effective Rights Results for Specific Attributes

```
# ldapsearch -D "cn=Directory Manager" -W -b "uid=scarter,ou=people,dc=example,dc=com" -E '!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" cn  
mail initials  
  
dn: uid=scarter,ou=People,dc=example,dc=com  
cn: Sam Carter  
mail: scarter@example.com  
entryLevelRights: vadn  
attributeLevelRights: cn:rscwo, mail:rscwo, initials:rscwo
```

It is possible to specify a non-existent attribute in the *attributeList*, as with the **initials** attribute in [Example 18.41, "Get Effective Rights Results for Specific Attributes"](#), to see the rights which are available, similar to using an asterisk to list all attributes.

The Directory Manager can also list the rights for all of the attributes available to a specific object class. This option has the format *attribute@objectClass*. This returns two entries; the first for the specified GER subject and the second for a template entry for the object class.

#### Example 18.42. Get Effective Rights Results for an Attribute within an Object Class

```
# ldapsearch -D "cn=Directory Manager" -W -b "uid=scarter,ou=people,dc=example,dc=com" -E '!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"  
uidNumber@posixAccount  
...  
dn: cn=template_posixaccount_objectclass,uid=scarter,ou=people,dc=example,dc=com  
uidnumber: (template_attribute)  
entryLevelRights: v  
attributeLevelRights: uidNumber:rsc
```



#### NOTE

Using the search format *attribute@objectClass* is only available if the requester (**-D**) is the Directory Manager.

Using an asterisk (\*) instead of a specific attribute returns all of the attributes (present and non-existent) for the specified GER subject and the full list of attributes for the object class template.

#### Example 18.43. Get Effective Rights Results for All Attributes for an Object Class

```
# ldapsearch -D "cn=Directory Manager" -W -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
*@posixaccount
...
dn: cn=template_posixaccount_objectclass,uid=scarter,ou=people,dc=example,dc=com
objectClass: posixaccount
objectClass: top
homeDirectory: (template_attribute)
gidNumber: (template_attribute)
uidNumber: (template_attribute)
uid: (template_attribute)
cn: (template_attribute)
entryLevelRights: v
attributeLevelRights: cn:rsc, uid:rsc, uidNumber:rsc, gidNumber:rsc, homeDirectory:rsc,
objectClass:rsc, userPassword:none, loginShell:rsc, gecos:rsc, description:rsc, aci:rsc
```

#### 18.11.3.4. Examples of Get Effective Rights Searches for Non-Existent Entries

An administrator may want to check what rights a specific user (**jsmith**) would have to a non-existent user, based on the existing access control rules. For checking non-existent entries, the server generates a template entry within that subtree. For example, to check for the template entry **cn=joe new user,cn=accounts,ou=people,dc=example,dc=com**, the server creates **cn=template,cn=accounts,ou=people,dc=example,dc=com**.

For checking a non-existent entry, the get effective rights search can use a specified object class to generate a template entry with all of the potential attributes of the (non-existent) entry. For **cn=joe new user,cn=accounts,ou=people,dc=example,dc=com** with a **person** object class (@**person**), the server generates **cn=template\_person\_objectclass,cn=accounts,ou=people,dc=example,dc=com**.

When the server creates the template entry, it uses the first MUST attribute in the object class definition to create the RDN attribute (or it uses MAY if there is no MUST attribute). However, this may result in an erroneous RDN value which, in turn, violates or circumvents established ACIs for the given subtree. In that case, it is possible to specify the RDN value to use by passing it with the object class. This has the form @*objectclass*:*rdn\_attribute*.

For example, to check the rights of **scarter** for a non-existent Posix entry with **uidNumber** as its RDN:

```
# ldapsearch -D "cn=Directory Manager" -W -b "ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
@posixaccount:uidnumber

dn: uidNumber=template_posixaccount_objectclass,ou=people,dc=example,dc=com
entryLevelRights: v
attributeLevelRights: description:rsc, gecos:rsc, loginShell:rsc, userPassword
:rsc, objectClass:rsc, homeDirectory:rsc, gidNumber:rsc, uidNumber:rsc, uid:
rsc, cn:rsc
```

#### 18.11.3.5. Examples of Get Effective Rights Searches for Operational Attributes

Operational attributes are not returned in regular **ldapsearches**, including get effective rights searches. To return the information for the operational attributes, use the plus sign (+). This returns only the operational attributes that can be used in the entry.

#### Example 18.44. Get Effective Rights Results for Operational Attributes

```
# ldapsearch -D "cn=Directory Manager" -W -x -b "uid=scarter,ou=people,dc=example,dc=com" -E '!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "+"

dn: uid=scarter,ou=People,dc=example,dc=com
entryLevelRights: vadn
attributeLevelRights: nsICQStatusText:rscwo, passwordGraceUserTime:rscwo,
pwdGraceUserTime:rscwo, nsYIMStatusText:rscwo, modifyTimestamp:rscwo,
passwordExpWarned:rscwo, pwdExpirationWarned:rscwo, entrydn:rscwo, aci:rscwo,
nsSizeLimit:rscwo, nsAccountLock:rscwo, passwordExpirationTime:rscwo, entryid:rscwo,
nsSchemaCSN:rscwo, nsRole:rscwo, retryCountResetTime:rscwo, ldapSchemas:rscwo,
nsAIMStatusText:rscwo, copiedFrom:rscwo, nsICQStatusGraphic:rscwo, nsUniqueId:rscwo,
creatorsName:rscwo, passwordRetryCount:rscwo, dncomp:rscwo, nsTimeLimit:rscwo,
passwordHistory:rscwo, pwdHistory:rscwo, nsCPEntryDN:rscwo, subschemaSubentry:rscwo,
nsYIMStatusGraphic:rscwo, hasSubordinates:rscwo, pwdPolicySubentry:rscwo,
nsAIMStatusGraphic:rscwo, nsRoleDN:rscwo, createTimeStamp:rscwo,
accountUnlockTime:rscwo, copyingFrom:rscwo, nsLookThroughLimit:rscwo,
nsds5ReplConflict:rscwo, modifiersName:rscwo, parentid:rscwo,
passwordAllowChangeTime:rscwo, nsBackendSuffix:rscwo, nsIdleTimeout:rscwo,
ldapSyntaxes:rscwo, numSubordinates:rscwo
```

#### 18.11.3.6. Examples of Get Effective Rights Results and Access Control Rules

Get effective rights are returned according to whatever ACLs are in effect for the get effective rights subject entry.

For example, this ACL is set and, for the purposes of this example, it is the only ACL set:

```
dn: dc=example,dc=com
objectClass: top
objectClass: domain
dc: example
aci: (target=ldap://ou=Accounting,dc=example,dc=com)(targetattr="*")(version
3.0; acl "test acl"; allow (read,search,compare) (userdn = "ldap://anyone") ;)

dn: ou=Accounting,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Accounting
```

Because the ACL does not include the **dc=example,dc=com** subtree, the get effective rights search shows that the user does not have any rights to the **dc=example,dc=com** entry:

#### Example 18.45. Get Effective Rights Results with No ACL Set (Directory Manager)

```
# ldapsearch -D "cn=Directory Manager" -W -b "dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
```

```
**@person

dn: cn=template_person_objectclass,uid=scarter,ou=people,dc=example,dc=com
objectClass: person
objectClass: top
cn: (template_attribute)
sn: (template_attribute)
description: (template_attribute)
seeAlso: (template_attribute)
telephoneNumber: (template_attribute)
userPassword: (template_attribute)
entryLevelRights: none
attributeLevelRights: sn:none, cn:none, objectClass:none, description:none, seeAlso:none,
telephoneNumber:none, userPassword:none, aci:none
```

If a regular user, rather than Directory Manager, tried to run the same command, the result would simply be blank.

#### Example 18.46. Get Effective Rights Results with No ACL Set (Regular User)

```
# ldapsearch -D "uid=scarter,ou=people,dc=example,dc=com" -W -b "dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2:=dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
**@person"
```

#### 18.11.4. Get Effective Rights Return Codes

If the criticality is not set 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 18.4, "Returned Result Codes"](#) summarizes the error codes and the potential configuration information they can relay.

Table 18.4. Returned Result Codes

Code	Description
0	Successfully completed.
1	Operation error.
12	The critical extension is unavailable. If the criticality expression is set to <b>true</b> 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.

Code	Description
50	Insufficient rights.
52	Unavailable.
53	Unwilling to perform.
80	Other.

## 18.12. LOGGING ACCESS CONTROL INFORMATION

To log access control information, set the ***nsslapd-errorlog-level*** parameter to a value that includes **128** (access control list processing). For further details about setting the error log level, see [Section 21.3.7, “Configuring the Log Levels”](#).

## 18.13. ADVANCED ACCESS CONTROL: USING MACRO ACIS

Macro ACIs improve the flexibility. For example, you can add a subtree and automatically get the same tailored access controls as for other subtrees without the need to add any ACI. As a side effect, the number of ACIs is smaller, however, Macro ACI processing is more expensive than a regular ACI.

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.

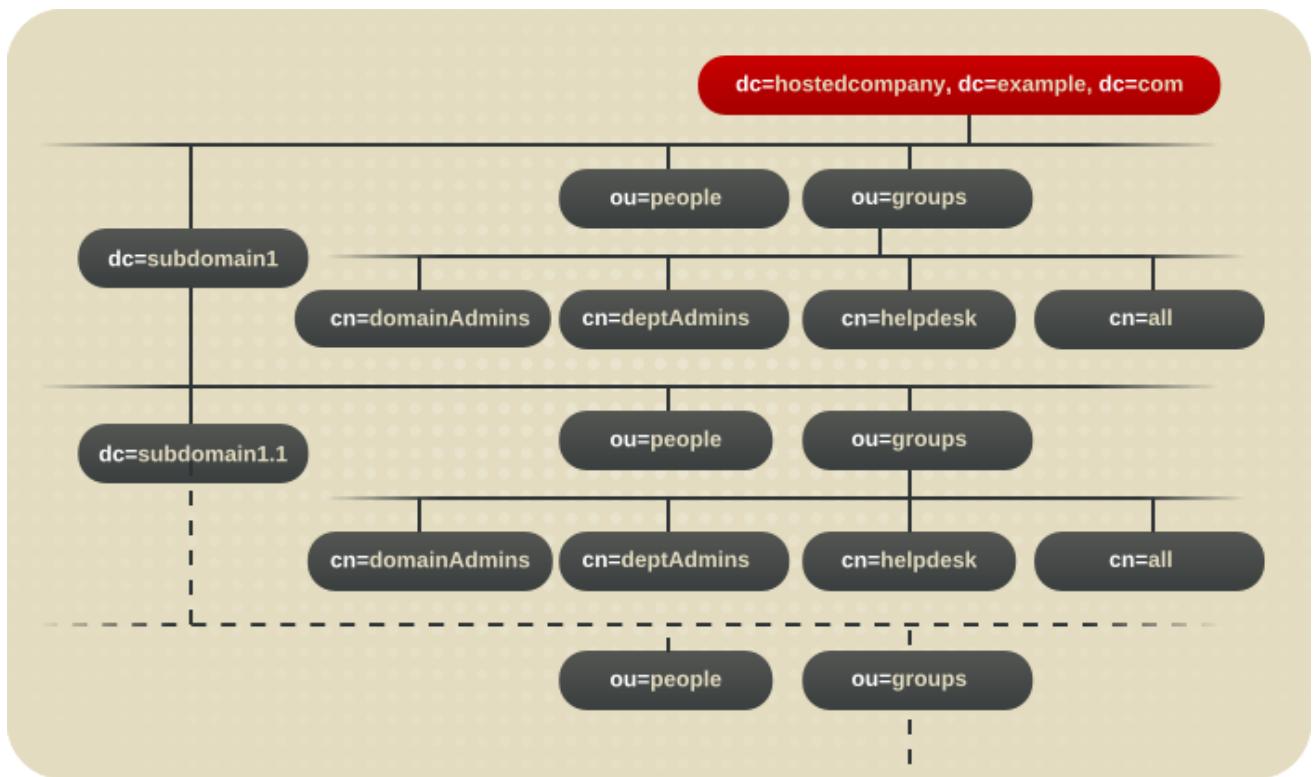
### 18.13.1. Macro ACI Example

[Figure 18.1, “Example Directory Tree for Macro ACIs”](#) shows a directory tree which uses macro ACIs to effectively reduce 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 Corp. 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 18.1. Example Directory Tree for Macro ACIs**

The following ACL 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 ACL 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 ACL 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 ACL 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, is utilized in the new ACI.

In this example, the number of ACIs is reduced from four to one. The real benefit is a factor of how many repeating patterns you have down and across your directory tree.

### 18.13.2. 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

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 18.5, “Macros in ACI Keywords”](#) shows in what parts of the ACI you can use DN macros:

**Table 18.5. Macros in ACI Keywords**

Macro	ACI Keyword
<b>(\$dn)</b>	target, targetfilter, userdn, roledn, groupdn, userattr
<b>[\$dn]</b>	targetfilter, userdn, roledn, groupdn, userattr
<b>(\$attr.attrName)</b>	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)**.



#### NOTE

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.

### 18.13.2.1. 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 is then 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.

### 18.13.2.2. 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 the target matches **dc=subdomain1,dc=hostedCompany1**.
2. [**\$dn**] in the subject is replaced with **dc=subdomain1,dc=hostedCompany1**.

The result is

**groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,dc=hostedCompany1,d  
c=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. [**\$dn**] in the subject is replaced 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=com");)
```

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 a subtree like **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.

### 18.13.2.3. Macro Matching for **(\$attr.attrName)**

The **(\$attr.attrName)** macro is always used in the subject part of a DN. For example, 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. For 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"
```

## 18.14. SETTING ACCESS CONTROLS ON DIRECTORY MANAGER

Having an unconstrained administrative user makes sense from a maintenance perspective. The Directory Manager requires a high level of access in order to perform maintenance tasks and to response to incidents.

However, because of the power of the Directory Manager user, a certain level of access control may be advisable to prevent unauthorized access or attacks from being performed as the root user.

Regular access control rules are applied to the directory tree, the Directory Manager is not a regular user entry, so no (regular) ACIs can be applied to the Directory Manager user. ACIs are applied through a special plug-in configuration entry.

### 18.14.1. About Access Controls on the Directory Manager Account

Normal access control rules do not apply to the Directory Manager user. The privileges of the Directory Manager user are hard-coded in Directory Server and cannot be used in a bind rule.

Access controls for Directory Manager are implemented through the *RootDN Access Control Plug-in*. This plug-in applies to the Directory Server configuration, and therefore can apply some access control rules to the Directory Manager entry.

The plug-in does not define a standard ACL. Some information is already implied, including the target (the Directory Manager entry) and the allowed rights (all of them). The purpose of the RootDN Access Control Plug-in is not to restrict *what* the Directory Manager can do; the purpose is to provide a level of security by limiting who can log in as Directory Manager (even with valid credentials) based on their location or time.

For this reason, the ACI for the Directory Manager only sets bind rules:

- Time-based access controls for time ranges, such as 8a.m. to 5p.m. (0800 to 1700), and day-of-week access controls, so access is only allowed on explicitly defined days. This is analogous to [Section 18.10.2.5, “Defining Access at a Specific Day of the Week”](#) and [Section 18.10.2.6, “Defining Access at a Specific Time of Day”](#).
- IP address rules, where only specified IP addresses, domains, or subnets are explicitly allowed or denied. This is analogous to [Section 18.10.2.2, “Defining Access from Specific IP Addresses or Ranges”](#).
- Host access rules, where only specified host names, domain names, or subdomains are explicitly allowed or denied. This is analogous to [Section 18.10.2.3, “Defining Access from a Specific Host or Domain”](#).

As with other access control rules, deny rules supercede allow rules.



## IMPORTANT

Make sure that the Directory Manager always has the appropriate level of access allowed. The Directory Manager may need to perform maintenance operations in off-hours (when user load is light) or to respond to failures. In that case, setting stringent time or day-based access control rules could prevent the Directory Manager from being able to adequately manage the directory.

### 18.14.2. Configuring the RootDN Access Control Plug-in

Root DN access control rules are disabled by default. Enable the **RootDN Access Control** plug-in, and then set the appropriate access control rules.



## NOTE

There is only one access control rule set for the Directory Manager, in the plug-in entry, and it applies to all access to the entire directory.

1. Enable the **RootDN Access Control** plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin root-dn enable
Plugin 'RootDN Access Control' enabled
...
```

2. Set the bind rules for the access control instruction. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin root-dn set --open-
time=0600 --close-time=2100 --allow-host="*.example.com" --deny-
host="*.remote.example.com"
```

You can set the following parameters:

- **--open-time** and **--close-time** for time-based access controls.
- **--days-allowed** for day-based access controls.
- **--allow-host**, **--deny-host**, **--allow-ip**, and **--deny-ip** for host-based access controls. These are all multi-valued attributes and you can use wild cards to allow or deny IP ranges or domains.



## IMPORTANT

Deny rules have a higher priority than allow rules. For example, if the **--allow-host** parameter is set to **\*.example.com**, and **--deny-host** is set to **\*.front-office.example.com**, access from all hosts in the **front-office.example.com** subdomain as Directory Manager is prevented.

3. Restart Directory Server:

```
# dsctl instance_name restart
```

# CHAPTER 19. USING THE HEALTH CHECK FEATURE TO IDENTIFY PROBLEMS

The **dsctl healthcheck** command analyzes the Directory Server instance for potential issues and recommends solutions to solve them.

The following table displays the checks the health check feature performs:

**Table 19.1. Overview of Checks**

Component	Severity	Result Code	Description
Backend	Low	DSBLE0003	The database was not initialized. A database was created but the database is empty.
Backend	Medium	DSBLE0001	The mapping tree entry for a back end is missing in the configuration.
Config	Low	DSCLE0001	High-resolution time stamps are disabled.
Config	High	DSVIRTLE0001	A virtual attribute is incorrectly indexed. Indexed attributes used by roles or Class of Service (CoS) definitions can corrupt search results.
Operating System	Medium	DSPERMLE0001	The permissions set on the <b>/etc/resolve.conf</b> file are different to <b>0644</b> .
Operating System	High	DSDSLE0001	Low disk space
Operating System	High	DSPERMLE0002	The permissions set on the <b>/etc/dirsrv/slapd-<i>instance_name</i>/pin.txt</b> and <b>/etc/dirsrv/slapd-<i>instance_name</i>/pwdfile.txt</b> files are different to <b>0400</b> .
Plug-ins	Low	DSRILE0001	An update delay is set for the <b>Referential Integrity</b> plug-in. This can cause replication issues.
Plug-ins	High	DSRILE0002	The <b>Referential Integrity</b> plug-in misses indexes. The plug-in queries certain attributes for every delete operation if they are not indexed. This can cause hard-to-detect unindexed searches and high CPU usage.
Replication	Low	DSREPLLE0002	Conflict entries exist in the database.
Replication	Low	DSSKEWLE0001	The replication time skew is larger than 6 hours and lower than 12 hours.

Component	Severity	Result Code	Description
Replication	Medium	DSCLLE0001	Changelog trimming is disabled. In this case, the changelog grows without limits.
Replication	Medium	DSREPLLE0004	The health check failed to retrieve the replication status.
Replication	Medium	DSREPLLE0003	The topology is not in sync, but the replication is working.
Replication	Medium	DSREPLLE0005	A remote replica is not reachable.
Replication	Medium	DSSKEWLE0002	The replication time skew is larger than 12 hours and lower than 24 hours.
Replication	High	DSREPLLE0001	The topology is not in sync, and the replication is not working.
Replication	High	DSSKEWLE0003	The replication time skew is larger than 24 hours. Replication sessions could break.
Security	Medium	DSELE0001	The minimum TLS version is set to a value lower than TLS 1.2.
Security	High	DSCLE0002	A weak password storage scheme is configured.
Server	High	DSBLE0002	The health check failed to query the back end.
TLS certificates	Medium	DSCRTLE0001	The server certificate expires within the next 30 days.
TLS certificates	High	DSCRTLE0002	The server certificate has expired.

## 19.1. RUNNING THE DIRECTORY SERVER HEALTH CHECK

To run the health check, enter:

```
# dsctl instance_name healthcheck
Beginning lint report, this could take a while ...
Checking Backends ...
Checking Config ...
Checking Encryption ...
Checking FSChecks ...
Checking ReferentialIntegrityPlugin ...
Checking MonitorDiskSpace ...
Checking Replica ...
Checking Changelog5 ...
```

```
Checking NSSSSL ...
Healthcheck complete.
1 Issue found! Generating report ...
```

### Example 19.1. Possible Report of the Health Check

The following shows an example health check report:

```
[1] DS Lint Error: DSELE0001
```

-----  
Severity: MEDIUM

Affects:

-- cn=encryption,cn=config

Details:

-----  
This Directory Server may not be using strong TLS protocol versions. TLS1.0 is known to have a number of issues with the protocol. Please see:

<https://tools.ietf.org/html/rfc7457>

It is advised you set this value to the maximum possible.

Resolution:

-----  
There are two options for setting the TLS minimum version allowed. You, can set "sslVersionMin" in "cn=encryption,cn=config" to a version greater than "TLS1.0" You can also use 'dsconf' to set this value. Here is an example:

```
# dsconf slapd-instance_name security set --tls-protocol-min=TLS1.2
```

You must restart the Directory Server for this change to take effect.

Or, you can set the system wide crypto policy to FUTURE which will use a higher TLS minimum version, but doing this affects the entire system:

```
# update-crypto-policies --set FUTURE
```

===== End Of Report (1 Issue found) =====

To display the output in JSON format, pass the **--json** parameter to the command:

```
# dsctl --json instance_name healthcheck
```

### Example 19.2. Possible Report of the Health Check in JSON Format

The following shows an example health check report in JSON format:

```
[
  {
    "dsle": "DSELE0001",
    "severity": "MEDIUM",
```

```
"items": [
    "cn=encryption,cn=config"
],
"detail": "This Directory Server may not be using strong TLS protocol versions. TLS1.0 is
known to\nhave a number of issues with the protocol. Please
see:\n\nhttps://tools.ietf.org/html/rfc7457\n\nIt is advised you set this value to the maximum
possible.",
"fix": "There are two options for setting the TLS minimum version allowed. You,\ncan set
\"sslVersionMin\" in \"cn=encryption,cn=config\" to a version greater than \"TLS1.0\"\nYou can also
use 'dsconf' to set this value. Here is an example:\n\n  # dsconf slapd-instance_name security
set --tls-protocol-min=TLS1.2\n\nYou must restart the Directory Server for this change to take
effect.\n\nOr, you can set the system wide crypto policy to FUTURE which will use a higher
TLS\nminimum version, but doing this affects the entire system:\n\n  # update-crypto-policies --
set FUTURE"
}
]
```

# CHAPTER 20. MANAGING USER AUTHENTICATION

When a user connects to the Red Hat Directory Server, first the user is authenticated. Then, the directory grants access rights and resource limits to the user depending upon the identity established during authentication.

This chapter describes tasks for managing users, including configuring the password and account lockout policy for the directory, denying groups of users access to the directory, and limiting system resources available to users depending upon their bind DNs.

## 20.1. SETTING USER PASSWORDS

You can use an entry 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, the user passwords can be set or reset with any LDAP operation, such as using the **ldapmodify** utility.

When an administrator changes the password of a user, Directory Server sets the ***pwdReset*** operational attribute in the user's entry to **true**. Applications can use this attribute to identify if a password of a user has been reset by an administrator.

For information on creating and modifying directory entries, see [Chapter 3, Managing Directory Entries](#). For information on inactivating user accounts, see [Section 20.16, “Manually Inactivating Users and Roles”](#).

Only password administrators, described in [Section 20.2, “Setting Password Administrators”](#), and the root DN can add pre-hashed passwords. These users can also violate password policies.



### WARNING

When using a password administrator account or the **Directory Manager** (root DN) to set a password, password policies are bypassed and not verified. Do not use these accounts for regular user password management. Use them only to perform password administration tasks that require bypassing the password policies.

## 20.2. SETTING PASSWORD ADMINISTRATORS

The Directory Manager can add the *password administrator* role to a user or a group of users. Since access control instructions (ACI) need to be set, it is recommended that a group is used to allow just a single ACI set to manage all password administrators. A password administrator can perform any user password operations, including the following:

- forcing the user to change their password,
- changing a user's password to a different storage scheme defined in the password policy,
- bypassing the password syntax checks,
- and adding already hashed passwords.

As explained in [Section 20.1, “Setting User Passwords”](#), it is recommended that ordinary password updates are done by an existing role in the database with permissions to update only the **userPassword** attribute. Red Hat recommends not to use the password administrator account for these ordinary tasks.

You can specify a user or a group as password administrator:

- In a local policy. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set
ou=people,dc=example,dc=com --pwdadmin
"cn=password_admins,ou=groups,dc=example,dc=com"
```

- In a global policy. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdadmin
"cn=password_admins,ou=groups,dc=example,dc=com"
```

## 20.3. CHANGING PASSWORDS STORED EXTERNALLY

While most passwords can be changed through **ldapmodify** operations, 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, so users can change their passwords, using a suitable client, in a standards-compliant way. The **dsidm** utility passes the changes for the password for the specified user:

```
# dsidm ldap://server.example.com -D bind_dn -W -b dc=example,dc=com account
change_password user newPassword oldPassword
```



### IMPORTANT

Password operations must be performed over a secure connection, meaning SASL, TLS, or STARTTLS. For information on using secure connections with LDAP client tools, see [Section 9.9.4, “Authenticating Using a Certificate”](#).

For further details about the parameters, see the output of the **dsidm instance\_name account change\_password --help** command.

To use STARTTLS, which runs the command on a non-secure port, run **dsidm** with the **-Z** option and the standard LDAP port number. The password extended change operation has the following format:

```
# dsidm ldap://server.example.com -Z bind_dn -W -b dc=example,dc=com account
change_password user newPassword oldPassword
```



### NOTE

For STARTTLS connections to work, the TLS environment variables must be configured as described in [Section 9.9.4, “Authenticating Using a Certificate”](#).

Use the **-Z** option to force the connection to be successful.

To modify an entry's password, run **dsidm** like any other operation. It is necessary to specify a bind DN, even if the account is the same as that given in the bind DN. For example:

```
# dsidm ldap://server.example.com -Z bind_dn -W -b dc=example,dc=com account
change_password user newPassword oldPassword
```

Access control is enforced for the password change operation. If the bind DN does not have rights to change the specified password, the operation will fail with an **Insufficient rights** error.

## 20.4. MANAGING THE PASSWORD POLICY

A password policy minimizes the risks of using passwords by enforcing a certain level of security. For example, a password policy can define that:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords.
- The password syntax must meet certain complexity requirements.



### WARNING

When using a password administrator account or the **Directory Manager** (root DN) to set a password, password policies are bypassed and not verified. Do not use these accounts for regular user password management. Use them only to perform password administration tasks that require bypassing the password policies.

Directory Server supports fine-grained password policy, so password policies 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).

The complete password policy applied to a user account is comprised of the following elements:

- *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 policies work in an inverted pyramid, from general to specific. A global password policy is superseded by a subtree-level password policy, which is superseded by a user-level password policy. Only one password policy is enforced for the entry; password policies are not additive. This means that if a particular attribute is configured in the global or subtree-level policy, but not in the user-level password policy, the attribute is not used for the user when a login is attempted because the active, applied policy is the user-level policy.
- *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.

**NOTE**

After establishing a password policy, user passwords can be protected 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.

### 20.4.1. Configuring the Global Password Policy

By default, global password policy settings are disabled. This section provides some examples how to configure a global password policy.

**NOTE**

After configuring the password policy, configure an account lockout policy. For details, see [Section 20.9, "Configuring a Password-Based Account Lockout Policy"](#).

#### 20.4.1.1. Configuring a Global Password Policy Using the Command Line

Use the **dsconf** utility to display and edit the global password policy settings:

1. Display the current settings:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy get
Global Password Policy: cn=config
-----
passwordstoragescheme: PBKDF2_SHA256
passwordChange: on
passwordMustChange: off
passwordHistory: off
passwordInHistory: 6
...
...
```

2. Adjust the password policy settings. For example, to enable the password syntax check and set the minimum length of passwords to **12** characters, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdchecksyntax=on --pwdmintokenlen=12
```

For a full list of available settings, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --help
```

3. Enable the password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdlockout on
```

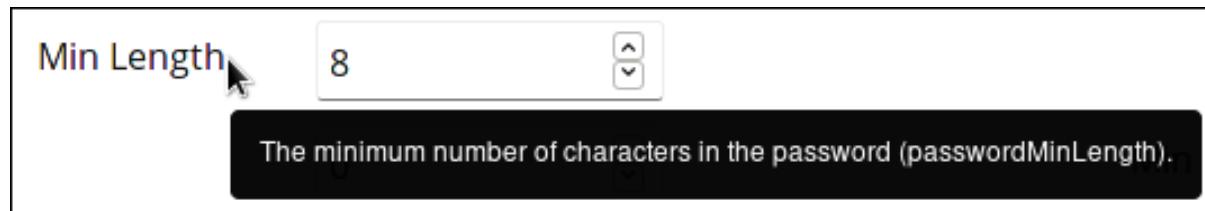
#### 20.4.1.2. Configuring a Global Password Policy Using the Web Console

To configure a global password policy using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).

2. Select the instance.
3. Open the **Database** menu.
4. In the **Password Policies** menu, select **Global Policy**.
5. Set the global password policy settings. You can set parameters in the following categories:
  - General settings, such as the password storage scheme
  - Password expiration settings, such as the time when a password expires.
  - Account lockout settings, such as after how many failed login attempts an account should be locked.
  - Password syntax settings, such as the minimum password length.

To display a tool tip and the corresponding attribute name for a parameter, hover the mouse cursor over the setting. For further details, see the parameter's description in the [Red Hat Directory Server Configuration, Command, and File Reference](#).



6. Click **Save**.

## 20.4.2. Using Local Password Policies

In contrast to a global password policy, which defines settings for the entire directory, a local password policy is a policy for a specific user or subtree.

### 20.4.2.1. Where Directory Server Stores Local Password Policy Entries

When you use the **dsconf localpwp adduser** or **dsconf localpwp addsubtree** commands, Directory Server creates automatically an entry to store the policy attributes:

- 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=newpwdpolicy_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=user\_name,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=user_name,ou=people,dc=example,dc=com",
cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy

```

#### 20.4.2.2. Configuring a Local Password Policy

To configure a local password policy:



## NOTE

Currently, you can only set up a local password policy using the command line.

- Verify if a local password policy already exists for the subtree or user entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp get  
"ou=People,dc=example,dc=com"  
Error: The policy wasn't set up for the target dn entry or it is invalid
```

If no local policy exists, create one:

- To create a subtree password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp addsubtree  
"ou=People,dc=example,dc=com"
```

- To create a user password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp adduser  
"uid=user_name,ou=People,dc=example,dc=com"
```

## IMPORTANT

When you create a new local policy, the previous commands automatically sets the ***nsslapd-pwpolicy-local*** parameter in the ***cn=config*** entry to **on**.

If the local password policy should not be enabled, manually set the parameter to **off**:

```
dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --  
pwdlocal off
```

- Set local policy attributes. For example, to enable password expiration and set the maximum password age to 14 days (**1209600** seconds):

- On a subtree password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --  
pwdexpire=on --pwdmaxage=1209600 "ou=People,dc=example,dc=com"
```

- On a user password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --  
pwdexpire=on --pwdmaxage=1209600  
"uid=user_name,ou=People,dc=example,dc=com"
```

For a full list of available settings, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --help
```

## 20.5. CONFIGURING TEMPORARY PASSWORD RULES

Directory Server password policies support setting temporary passwords on user accounts. If you assign a temporary password to a user, Directory Server rejects any other operation than a password change until the user changes its password.

The following are the features of temporary passwords:

- Only the **cn=Directory Manager** account can assign temporary passwords.
- Directory Server allows authentication attempts only for a fixed number of times to avoid that an attacker probes the password.
- Directory Server allows authentication attempts after a specified delay to configure that the temporary passwords are not usable directly after you set them.
- Directory Server allows authentication attempts only for a specified time so that the temporary password expires if a user does not use or reset it.
- If the authentication was successful, Directory Server requires that the user resets the password before the server performs any other operation.

By default, temporary password rules are disabled. You can configure them in global or local password policies.

### 20.5.1. Enabling temporary password rules in the global password policy

To enable the temporary password feature for the whole Directory Server instance:

1. Enable that users must change their password if an administrator resets it.
2. Configure the feature in the global password policy.

If an administrator updates the **userPassword** attribute of a user and sets the **passwordMustChange** attribute to **on**, Directory Server applies the temporary password rules.

#### Procedure

1. Configure that a user must change its password after an administrator resets it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdmustchange on
```

2. Configure the temporary password rules settings in a global password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60
```

In this example:

- The **--pwptprmaxuse** option sets the maximum number of attempts a user can use the temporary password to **5**.
- The **--pwptprdelayexpireat** option sets the time before the temporary password expires to **3600** seconds (1 hour)

- The **--pwptprdelayvalidfrom** option configures that the time set in **--pwptprdelayexpireat** starts **60** seconds after an administrator reset the password of a user.

## Verification

- Display the attributes that store the temporary password rules:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy get | grep -i TPR
passwordTPRMaxUse: 5
passwordTPRDelayExpireAt: 3600
passwordTPRDelayValidFrom: 60
```

### 20.5.2. Enabling temporary password rules in a local password policy

To enable the temporary password feature for a specific user or sub-tree, enable that users must change their password if an administrator resets it, and configure the feature in a local password policy.

If an administrator updates the ***userPassword*** attribute of a user and sets the ***passwordMustChange*** attribute to **on**, Directory Server applies the temporary password rules if the user:

- Has the local password policy enabled
- Is stored in a sub-tree that has the local password policy enabled

## Procedure

1. Configure that a user must change its password after an administrator resets it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdmustchange on
```

2. Configure the temporary password rules settings:

- For a sub-tree:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp addsubtree --
pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60
ou=People,dc=example,dc=com
```

- For a user:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp adduser --
pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60
uid=example,ou=People,dc=example,dc=com
```

Note that you can only set a local password policy on entries that exist.

In these examples:

- The **--pwptprmaxuse** option sets the maximum number of attempts a user can use the temporary password to **5**.
- The **--pwptprdelayexpireat** option sets the time before the temporary password expires to **3600** seconds (1 hour).

- The **-pwptprdelayvalidfrom** option configures that the time set in **--pwptprdelayexpireat** starts **60** seconds after an administrator reset the password of a user.

## Verification

- Display the local password policy of the distinguished name (DN):

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp get
distinguished_name | grep -i TPR
passwordTPRMaxUse: 5
passwordTPRDelayExpireAt: 3600
passwordTPRDelayValidFrom: 60
```

## 20.6. UNDERSTANDING PASSWORD EXPIRATION CONTROLS

When a user authenticates to Directory Server using a valid password, and if the password is expired, will expire soon, or needs to be reset, the server sends the following LDAP controls back to the client:

- Expired control (**2.16.840.1.113730.3.4.4**): Indicates that the password is expired. Directory Server sends this control in the following situations:
  - The password is expired, and grace logins have been exhausted. The server rejects the bind with an **Error 49** message.
  - The password is expired, but grace logins are still available. The bind will be allowed.
  - If **passwordMustChange** is enabled in the **cn=config** entry, and a user needs to reset the password after an administrator changed it. The bind is allowed, but any subsequent operation, other than changing the password, results in an **Error 53** message.
- Expiring control (**2.16.840.1.113730.3.4.5**): Indicates that the password will expire soon. Directory Server sends this control in the following situations:
  - The password will expire within the password warning period set in the **passwordWarning** attribute in the **cn=config** entry.
  - If the password policy configuration option is enabled in the **passwordSendExpiringTime** attribute in the **cn=config** entry, the expiring control is always returned, regardless of whether the password is within the warning period.
- Bind response control (**1.3.6.1.4.1.42.2.27.8.5.1**): The control contains detailed information about the state of the password that is about to expire or will expire soon.



### NOTE

Directory Server only sends the bind response control if the client requested it. For example, if you use **ldapsearch**, you must pass the **-e ppolicy** parameter to the command to request the bind response control.

#### Example 20.1. Requesting the Bind Response Control in a Query

If you request the bind response control, for example by passing the **-e ppolicy** parameter to the **ldapsearch** command, the server returns detailed information about account expiration. For example:

```
# ldapsearch -D "uid=user_name,dc=example,dc=com" -xLLL -W \
-b "dc=example,dc=com" -e ppolicy
ldap_bind: Success (0); Password expired (Password expired, 1 grace logins remain)
```

## 20.7. MANAGING THE DIRECTORY MANAGER PASSWORD

The Directory Manager is the privileged database administrator, comparable to the **root** user in Linux. The Directory Manager entry and the corresponding password are set during the instance installation.

The default distinguished name (DN) of the Directory Manager is **cn=Directory Manager**.

### 20.7.1. Resetting the Directory Manager Password

If you lose the Directory Manager password, reset it:

1. Stop the Directory Server instance:

```
# dsctl instance_name stop
```

2. Generate a new password hash. For example:

```
# pwdhash -D /etc/dirsrv/slapd-instance_name password
{PBKDF2_SHA256}AAAgABU0bKhyjY53NcxY33ueoPjOUWtl4iyYN5uW...
```

Specifying the path to the Directory Server configuration automatically uses the password storage scheme set in the **nsslapd-rootpwstorage** attribute to encrypt the new password.

3. Edit the **/etc/dirsrv/slapd-instance\_name/dse.ldif** file and set the **nsslapd-rootpw** attribute to the value displayed in the previous step:

```
nsslapd-rootpw: {PBKDF2_SHA256}AAAgABU0bKhyjY53NcxY33ueoPjOUWtl4iyYN5uW...
```

4. Start the Directory Server instance:

```
# dsctl instance_name start
```

### 20.7.2. Changing the Directory Manager Password

This section describes how to can change the password of the Directory Manager account.

#### 20.7.2.1. Changing the Directory Manager Password Using the Command Line

Use one of the following options to set the new password:



## IMPORTANT

Only set the password using an encrypted connection. Using an unencrypted connection can expose the password to the network. If your server does not support encrypted connections, use the web console to update the Directory Manager password. See [Section 20.7.2.2, “Changing the Directory Manager Password Using the Web Console”](#).

- To set the ***nsslapd-rootpw*** parameter to a plain text value which Directory Server automatically encrypts:

```
# dsconf -D "cn=Directory Manager" ldaps://server.example.com config replace nsslapd-rootpw=password
```



## WARNING

Do not use curly braces ({} ) in the password. Directory Server stores the password in the **{password-storage-scheme}hashed\_password** format. The server interprets characters in curly braces as the password storage scheme. If the string is an invalid storage scheme or if the password is not correctly hashed, the Directory Manager cannot connect to the server.

- To manually encrypt the password and setting it in the ***nsslapd-rootpw*** parameter:

1. Generate a new password hash. For example:

```
# pwdhash -D /etc/dirsrv/slapd-instance_name password
{PBKDF2_SHA256}AAAgAMwPYlhEkQozTagoX6RGG5E7d6/6oOJ8TVty...
```

Specifying the path to the Directory Server configuration automatically uses the password storage scheme set in the ***nsslapd-rootpwstorageScheme*** attribute to encrypt the new password.

2. Set the ***nsslapd-rootpw*** attribute to the value displayed in the previous step using a secure connection (STARTTLS):

```
# dsconf -D "cn=Directory Manager" ldaps://server.example.com config replace nsslapd-rootpw="{PBKDF2_SHA256}AAAgAMwPYlhEkQozTagoX6RGG5E7d6/6oOJ8TVty..."
```

### 20.7.2.2. Changing the Directory Manager Password Using the Web Console

As the administrator, perform these steps to change the password:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select **Server Settings**.

4. Open the **Directory Manager** tab.
5. Enter the new password into the **Directory Manager Password** and **Confirm Password** fields
6. Optionally, set a different password storage scheme.
7. Click **Save**.

### 20.7.3. Changing the Directory Manager Password Storage Scheme

The password storage scheme specifies which algorithm Directory Server uses to hash a password. To change the storage scheme using the command line, your server must support encrypted connections. If your server does not support encrypted connections, use the web console to set the storage scheme. See [Section 20.7.3.2, “Changing the Directory Manager Password Storage Scheme Using the Web Console”](#).

Note that the storage scheme of the Directory Manager (**nsslapd-rootpwstoragescheme**) can be different than the scheme used to encrypt user passwords (**nsslapd-pwstoragescheme**).

For a list of supported password storage schemes, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).



#### NOTE

If you change the Directory Manager's password storage scheme you must also reset its password. Existing passwords cannot be re-encrypted.

#### 20.7.3.1. Changing the Directory Manager Password Storage Scheme Using the Command Line

If your server supports encrypted connections, perform these steps to change the password storage scheme:

1. Generate a new password hash that uses the new storage scheme. For example:

```
# pwdhash -s PBKDF2_SHA256 password
{PBKDF2_SHA256}AAAgAMwPYlhEkQozTagoX6RGG5E7d6/6oOJ8TVty...
```

2. Set the **nsslapd-rootpwstoragescheme** attribute to the storage scheme and the **nsslapd-rootpw** attribute to the value displayed in the previous step using a secure connection (STARTTLS):

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-
rootpwstoragescheme=PBKDF2_SHA256 nsslapd-
rootpw="{PBKDF2_SHA256}AAAgAMwPYlhEkQozTagoX6RGG5E7d6/6oOJ8TVty..."
```

#### 20.7.3.2. Changing the Directory Manager Password Storage Scheme Using the Web Console

Perform these steps to change the password using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).

2. Select the instance.
3. Open the **Server Settings** menu, and select **Server Settings**.
4. Open the **Directory Manager** tab.
5. Set the password storage scheme.
6. Directory Server cannot re-encrypt the current password using the new storage scheme. Therefore, enter a new password into the **Directory Manager Password** and **Confirm Password** field.
7. Click **Save Configuration**.

#### 20.7.4. Changing the Directory Manager DN

As the administrator, perform the following step to change the Directory Manager DN to **cn>New Directory Manager**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-rootdn="cn>New Directory Manager"
```

Note that Directory Server supports only changing the Directory Manager DNs using the command line.

## 20.8. CHECKING ACCOUNT AVAILABILITY FOR PASSWORDLESS ACCESS

Most of the time, for the Directory Server to return authentication information about a user account, a client actually binds (or attempts to bind) as that user. And a bind attempt requires some sort of user credentials, usually a password or a certificate. While the Directory Server allows unauthenticated binds and anonymous binds, neither of those binds returns any user account information.

There are some situations where a client requires information about a user account – specifically whether an account should be allowed to authenticate – in order to perform some other operation, but the client either does not have or does not use any credentials for the user account in Directory Server. Essentially, the client needs to perform a credential-less yet authenticated bind operation to retrieve the user account information (including password expiration information, if the account has a password).

This can be done through an **Idapsearch** by passing the *Account Usability Extension Control*. This control acts as if it performs an authenticated bind operation for a given user and returns the account status for that user – but without actually binding to the server. This allows a client to determine whether that account can be used to log in and then to pass that account information to another application, like PAM.

For example, using the Account Usability Extension Control can allow a system to use the Directory Server as its identity back end to store user data but to employ password-less authentication methods, like smart cards or SSH keys, where the authentication operation is performed outside Directory Server.

#### 20.8.1. Searching for Entries Using the Account Usability Extension Control

The Account Usability Extension Control is an extension for an **Idapsearch**. It returns an extra line for each returned entry that gives the account status and some information about the password policy for that account. A client or application can then use that status to evaluate authentication attempts made

outside Directory Server for that user account. Basically, this control signals whether a user should be allowed to authenticate without having to perform an authentication operation.



### NOTE

The OpenLDAP tools used by Directory Server do not support the Account Usability Extension Control. Other LDAP utilities, like OpenDS, can be used or other clients which do support the control.

For example, using the OpenDS tools, the control can be specified using the **-J** with the control OID (1.3.6.1.4.1.42.2.27.9.5.8) or with the **accountusability:true** flag:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -J "accountusability:true" "(objectclass=*)"
# Account Usability Response Control
# The account is usable
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example
...
...
```

This can also be run for a specific entry:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b
"uid=bjensen,ou=people,dc=example,dc=com" -s base -J "accountusability:true" "(objectclass=*)"
# Account Usability Response Control
# The account is usable
dn: uid=bjensen,ou=people,dc=example,dc=com
...
...
```



### NOTE

By default, only the Directory Manager can use the Account Usability Extension Control. To allow other users to use the Account Usability Extension Control, set on ACI on the supported control entry under **cn=features**. See [Section 20.8.2, “Changing What Users Can Perform an Account Usability Search”](#).

The control returns different messages, depending on the actual status of the account and (if the user has a password) the password policy settings for the user account.

**Table 20.1. Possible Account Usability Control Result Messages**

Account Status	Control Result Message
Active account with a valid password	The account is usable
Active account with no password set	The account is usable
Expired password	Password expired

Account Status	Control Result Message
The password policy for the account is modified	Password expired
The account is locked and there is no lockout duration	Password reset
The account is locked and there is a lockout duration	<i>Time</i> (in seconds) for automatic unlock of the account
The password for the account should be reset at the first login	Password reset
The password has expired and grace logins are allowed	Password expired and <i>X</i> grace login is allowed
The password has expired and the number of grace logins is exhausted	Password expired
The password will expire (expiration warning)	Password will expire in <i>X</i> number of seconds

### 20.8.2. Changing What Users Can Perform an Account Usability Search

By default, only the Directory Manager can use the Account Usability Extension Control. Other users can use the Account Usability Extension Control by setting the appropriate ACI on the supported control entry. The control entry is named for the Account Usability Extension Control OID, 1.3.6.1.4.1.42.2.27.9.5.8.

For example, to enable members of the **cn=Administrators,ou=groups,dc=example,dc=com** group to read the Account Usability Extension Control of all users:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: oid=1.3.6.1.4.1.42.2.27.9.5.8,cn=features,cn=config
changetype: modify
add: aci
aci: (targetattr = "")(version 3.0; acl "Account Usable"; allow (read)(groupdn =
"ldap://cn=Administrators,ou=groups,dc=example,dc=com");)
```

## 20.9. CONFIGURING A PASSWORD-BASED ACCOUNT LOCKOUT POLICY

A password-based account lockout policy protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. The password policy can be set so that a specific user is locked out of the directory after a given number of failed attempts to bind.

### 20.9.1. Configuring the Account Lockout Policy Using the Command Line

Use a **dsconf pwpolicy set** command to configure the account lockout policy settings. For example, to enable the lockout policy and configure that accounts are locked after four failed login attempts:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdlockout on --
pwdmaxfailures=4
```

The following parameters control the account password policy:

- **--pwdlockout**: Set this parameter to **on** or **off** to enable or disable the account lockout feature.
- **--pwdunlock**: Set this parameter to **on** to unlock an account after the lockout duration.
- **--pwdlockoutduration**: Sets the number of seconds for which an account will be locked out.
- **--pwdmaxfailures**: Sets the maximum number of allowed failed password attempts before the account gets locked.
- **--pwdresetfailcount**: Sets the number of seconds before Directory Server resets the failed login count of an account.

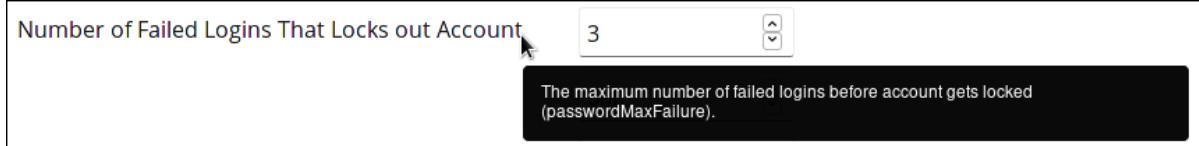
## 20.9.2. Configuring the Account Lockout Policy Using the Web Console

To configure the account lockout policy using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Database** tab, and select **Global Password Policy**.
4. On the **Account Lockout** tab, enable **Enable Account Lockout** setting and set the parameters. For example:

Global Password Policy			
General Settings	Expiration	Account Lockout	Syntax Checking
<input checked="" type="checkbox"/> <b>Enable Account Lockout</b>			
<b>Number of Failed Logins That Locks out Account</b>		<input type="text" value="3"/>	
<b>Time Until Failure Count Resets</b>		<input type="text" value="600"/>	
<b>Time Until Account Unlocked</b>		<input type="text" value="3600"/>	
<input checked="" type="checkbox"/> <b>Do Not Lockout Account Forever</b>			
<input type="button" value="Save"/>			

To display a tool tip and the corresponding attribute name in the **cn=config** entry for a parameter, hover the mouse cursor over the setting. For further details, see the parameter's description in the [Red Hat Directory Server Configuration, Command, and File Reference](#).



5. Click **Save**.

### 20.9.3. Disabling Legacy Password Lockout Behavior

There are different ways of interpreting when the maximum password failure (***passwordMaxFailure***) has been reached. It depends on how the server counts the last failed attempt in the overall failure count.

The traditional behavior for LDAP clients is to assume that the failure occurs *after* the limit has been reached. So, if the failure limit is set to three, then the lockout happens at the fourth failed attempt. This also means that if the fourth attempt is successful, then the user can authenticate successfully, even though the user technically hit the failure limit. This is  $n+1$  on the count.

LDAP clients increasingly expect the maximum failure limit to look at the last failed attempt in the count as the final attempt. So, if the failure limit is set to three, then at the third failure, the account is locked. A fourth attempt, even with the correct credentials, fails. This is  $n$  on the count.

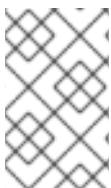
The first scenario – where an account is locked only if the attempt count is exceeded – is the historical behavior, so this is considered a legacy password policy behavior. In Directory Server, this policy is enabled by default, so an account is only locked when the failure count is  $n+1$ . This legacy behavior can be disabled so that newer LDAP clients receive the error (LDAP\_CONSTRAINT\_VIOLATION) when they expect it. This is set in the ***passwordLegacyPolicy*** parameter.

To disable the legacy password lockout behavior:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
passwordLegacyPolicy=off
```

## 20.10. CONFIGURING TIME-BASED ACCOUNT LOCKOUT POLICIES

Aside from locking accounts for failed authentication attempts, another method of defining an account lockout policy is to base it on account inactivity or an account age. The Account Policy Plug-in uses a *relative* time setting to determine whether an account should be locked.



### NOTE

Roles or classes of service can be used to inactivate accounts based on *absolute* account times. For example, a CoS can be created that inactivates every account created before a certain date.

The Account Policy Plug-in requires three configuration entries:

- A configuration entry for the plug-in itself. This sets global values that are used for all account policies configured on that server.
- An account policy configuration entry. This entry is within the user directory and is essentially a template which is referenced and applied to user account entries.

- An entry which applies the account policy entry. A user account can reference an account policy directly or a CoS or role can be used to apply account policies to sets of user accounts automatically.



### NOTE

An account policy is applied through the ***acctPolicySubentry*** attribute. While this attribute can be added directly to user accounts, this attribute is single-valued – which means that only one account policy can be applied to that account.

That may be fine in most cases. However, an organization could realistically create two account policies, one for account inactivity and then another for account expiration based on age.

Using a CoS to apply account policies allows multiple account policies to be used for an account.

#### 20.10.1. Account Policy Plug-in Syntax

The Account Policy Plug-in itself only has two configuration attributes:

- *nsslapd-pluginEnabled*, which sets whether the plug-in is enabled or disabled. This attribute is **off** by default.
- *nsslapd-pluginarg0*, which points to the DN of the plug-in configuration directory. The configuration entry is usually a child entry of the plug-in itself, such as **cn=config,cn=Account Policy Plugin,cn=plugins,cn=config**.

Past that, account policies are defined in two parts:

- The plug-in configuration entry identified in the *nsslapd-pluginarg0* attribute. This sets global configuration for the plug-in to use to identify account policy configuration entries and to manage user account entries. These settings apply across the server.

The configuration entry attributes are described in the [Account Policy Plug-in Attributes](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

- The account policy configuration entry. This is much like a template entry, which sets specific values for the account policies. User accounts – either directly or through CoS entries – reference this account policy entry.

The account policy and user entry attributes are described in the following table:

**Table 20.2. Account Policy Entry and User Entry Attributes**

Attribute	Definition	Configuration or User Entry
accountpolicy (object class)	Defines a template entry for account inactivation or expiration policies.	Configuration
accountInactivityLimit (attribute)	Sets the time period, in seconds, from the last login time of an account before that account is locked for inactivity.	Configuration

Attribute	Definition	Configuration or User Entry
acctPolicySubentry (attribute)	Identifies any entry which belongs to an account policy (specifically, an account lockout policy). The value of this attribute points to the DN of the account policy which is applied to the entry.	User
createTimestamp (operational attribute)	Contains the date and time that the entry was initially created.	User
lastLoginTime (operational attribute)	Contains a timestamp of the last time that the given account authenticated to the directory.	User

For further details, see the attribute's description in the [Red Hat Directory Server Configuration, Command, and File Reference](#)

### 20.10.2. Account Inactivity and Account Expiration

The **Account Policy** plug-in enables you to set up:

- account expiration: Accounts are disabled a certain amount of time after you created an account.
- account inactivity: Accounts are disabled a certain amount of time after the last successful login. This enables you to automatically disable unused accounts.

Disabled accounts are no longer able to log in.

To set up the **Account Policy** plug-in:

1. Enable the Account Policy Plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin account-policy enable
```

2. Set the plug-in configuration entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin account-policy set --config-entry="cn=config,cn=Account Policy Plugin,cn=plugins,cn=config"
```

3. Create the plug-in configuration entry:

- To use CoS or roles with account policies, set the **alwaysRecordLogin** value to **yes**. This means every entry has a login time recorded, even if it does not have the **acctPolicySubentry** attribute.
- Set the primary attribute to use for the account policy evaluation as value for **stateAttrName**. For account inactivity, use the **lastLoginTime** attribute. For a simple account expiration time, use **createTimestamp** attribute.

- You can set a secondary attribute in ***altStateAttrName***, that is checked if the primary one defined in ***stateAttrName*** does not exist. If no attribute is specified as alternative the default value ***createTimestamp*** is used.



### WARNING

If the value for the primary attribute is set to ***lastLoginTime*** and ***altStateAttrName*** to ***createTimestamp***, users in existing environments are automatically locked out when their accounts do not have the ***lastLoginTime*** attribute and the ***createTimestamp*** is older than the configured inactivity period.

To avert this situation, set the alternative attribute to **1.1**. This explicitly states to use no attribute as alternative. The ***lastLoginTime*** attribute will be created automatically after the user logs in the next time.

- Set the attribute to use to show which entries have an account policy applied to them (***acctPolicySubentry***).
- Set the attribute in the account policy which is used to set the actual timeout period, in seconds (***accountInactivityLimit***).

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin account-policy config-entry add "cn=config,cn=Account Policy Plugin,cn=plugins,cn=config" --always-record-login yes --state-attr lastLoginTime --alt-state-attr 1.1 --spec-attr acctPolicySubentry --limit-attr accountInactivityLimit
```

4. Restart the server to load the new plug-in configuration:

```
# dsctl instance_name restart
```

5. Define an account policy:

```
# ldapadd -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=Account Inactivation Policy,dc=example,dc=com

objectClass: top
objectClass: ldapsubentry
objectClass: extensibleObject
objectClass: accountpolicy
accountInactivityLimit: 2592000
cn: Account Inactivation Policy
```

6. Create the class of service template entry:

```
# ldapadd -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=TemptlCoS,dc=example,dc=com
```

```

objectClass: top
objectClass: ldapsubentry
objectClass: extensibleObject
objectClass: cosTemplate
acctPolicySubentry: cn=Account Inactivation Policy,dc=example,dc=com

```

Account policies can be defined directly on user entries, instead of using a CoS. However, using a CoS allows an account policy to be applied and updated reliably for multiple entries and it allows multiple policies to be applied to an entry.

7. Create the class of service definition entry. The managed entry for the CoS is the account policy attribute, ***acctPolicySubentry***. This example applies the CoS to the entire directory tree:

```

# ldapadd -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=DefnCoS,dc=example,dc=com

objectClass: top
objectClass: ldapsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=TempltCoS,dc=example,dc=com
cosAttribute: acctPolicySubentry default operational-default

```

### 20.10.3. Disabling Accounts a Certain Amount of Time After Password Expiry

Directory Server enables you to configure an account policy that disables an account a certain amount of time after the password expired. Disabled accounts are no longer able to log in.

To set up this configuration, follow the procedure in [Section 20.10.2, “Account Inactivity and Account Expiration”](#). However, when configuring the plug-in configuration entry, use the following settings instead:

```

dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

objectClass: top
objectClass: extensibleObject
cn: config
alwaysrecordlogin: yes
stateAttrName: non_existent_attribute
altStateAttrName: passwordExpirationTime
specattrname: acctPolicySubentry
limitattrname: accountInactivityLimit

```

This configuration uses a dummy value in the ***stateAttrName*** parameter. Therefore, only the ***passwordExpirationTime*** attribute set in the ***altStateAttrName*** parameter is used to calculate when an account is expired.

To additionally record the time of the last successful login in the ***lastLoginTime*** attribute of the user entry, set:

```

dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

alwaysRecordLoginAttr: lastLoginTime

```

Using this configuration, an account is automatically disabled if the sum of the time set in the user's **passwordExpirationTime** attribute and in the **accountInactivityLimit** parameter's value is in the past. Using this configuration, an account is automatically disabled if the sum of the value in the user's **passwordExpirationTime** attribute and in the **accountInactivityLimit** parameter exceeds the time since the **alwaysRecordLoginAttr** attribute was last updated.

#### 20.10.4. Tracking Login Times without Setting Lockout Policies

It is also possible to use the Account Policy Plug-in to track user login times *without* setting an expiration time or inactivity period. In this case, the Account Policy Plug-in is used to add the **lastLoginTime** attribute to user entries, but no other policy rules need to be set.

In that case, set up the Account Policy Plug-in as normal, to track login times. However, do not create a CoS to act on the login information that is being tracked.

1. Enable the Account Policy Plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin account-policy enable
```

2. Create the plug-in configuration entry to record login times:

- Set the **alwaysRecordLogin** value to yes so that every entry has a login time recorded.
- Set the **lastLoginTime** attribute as the attribute to use for the account policy (**stateattrname**).
- Set the attribute to use to show which entries have an account policy applied to them (**acctPolicySubentry**).
- Set the attribute in the account policy which is used to set the actual timeout period, in seconds (**accountInactivityLimit**).

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin account-policy config-entry set "cn=config,cn=Account Policy Plugin,cn=plugins,cn=config" --always-record-login yes --state-attr lastLoginTime --alt-state-attr createTimeStamp --spec-attr acctPolicySubentry --limit-attr accountInactivityLimit
```

3. Restart the server to load the new plug-in configuration:

```
# dsctl instance_name restart
```

#### 20.10.5. Unlocking Inactive Accounts

If an account is locked because it reached the inactivity limit, you can reactivate it using one of the following methods:

- Using the **dsidm** utility:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" account unlock "uid=example"
```

- Manually by resetting the **lastLoginTime** attribute to a current time stamp:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=example,ou=people,dc=example,dc=com
changetype: modify
replace: lastLoginTime
lastLoginTime: 20210901000000Z
```

The ***lastLoginTime*** attribute stores its value in GMT/UTC time (Zulu time zone), indicated by the appended **Z** to the time stamp.

## 20.11. REPLICATING ACCOUNT LOCKOUT ATTRIBUTES

Account lockout policies will block a user ID from being able to access the Directory Server if the login attempt fails a set number of times. This prevents hackers or other malicious people from illegitimately accessing the Directory Server by guessing a password. Password policies are set locally, and generally account lockout attributes are local to each replica. This means that a person can attempt to log in to one replica until the account lockout count is reached, then try again immediately on another replica. The way to prevent that is to replicate the attributes related to the account lockout counts for an entry, so that the malicious user is locked out of every supplier and consumer replica in the configuration if a login attempt fails on a single supplier.

By default, three password policy attributes are not replicated, even if other password attributes are. These attributes are related to of login failures and lockout periods:

- ***passwordRetryCount***
- ***retryCountResetTime***
- ***accountUnlockTime***

### 20.11.1. Managing the Account Lockouts and Replication

Password and account lockout policies are enforced in a replicated environment slightly differently:

- Password policies are enforced on the data supplier.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy information in the directory is replicated automatically:

- ***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, unless specifically configured for replication.

When configuring a password policy in a replicated environment, make sure that these elements are in place, so password policies and account lockout settings are enforced consistently:

- Warnings from the server of an impending password expiration are 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.

- The same bind behavior should occur on all servers, including suppliers and replicas. Make sure to create the same password policy configuration information on each server.
- Account lockout counters may not work as expected in a multi-supplier environment. Account lockout counters are not replicated by default (although this can be configured). If account lockout attributes are not replicated at all, then a user could be locked out from one server but could successfully bind to another server (or, conversely, a user may be unlocked on one server and still blocked on another). If account lockout attributes are replicated, then there could be lags between an account lockout change on one server and when that change is propagated to the other servers. It depends on the replication schedule.
- 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).



#### NOTE

If the password policy is enabled and the ***alwaysRecordLogin*** parameter set to **yes**, the value of the ***lastLoginTime*** attribute can be different on suppliers and read-only replicas. For example, if a user logs in to a read-only replica, the ***lastLoginTime*** attribute is updated locally but the value is not replicated to the supplier servers.

### 20.11.2. Configuring Directory Server to Replicate Password Policy Attributes

A special core configuration attribute controls whether password policy operational attributes are replicated. This is the ***passwordIsGlobalPolicy*** attribute, which is enabled in the consumer Directory Server configuration to allow the consumer to accept password policy operational attributes.

By default, this attribute is set to **off**.

To enable these attributes to be replicated, change the ***passwordIsGlobalPolicy*** configuration parameter on the consumer:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdisglobal="on"
```

Changing that value to **on** allows the ***passwordRetryCount***, ***retryCountResetTime***, and ***accountUnlockTime*** to be replicated.

### 20.11.3. Configuring Fractional Replication for Password Policy Attributes

Setting the ***passwordIsGlobalPolicy*** attribute affects the consumer in replication, in that it allows the consumer to receive updates to those attributes. To control whether the password policy attributes are actually replicated by the supplier, use fractional replication, which controls what specific entry attributes are replicated.

If the password policy attributes should be replicated, then make sure these attributes are included in the fractional replication agreement (as they are by default).

If the ***passwordIsGlobalPolicy*** attribute is set to **off** on the consumer, so no password policy attributes should be replicated, use fractional replication (described in [Section 15.1.7, “Replicating a Subset of Attributes with Fractional Replication”](#)) to enforce that on the supplier and specifically exclude those

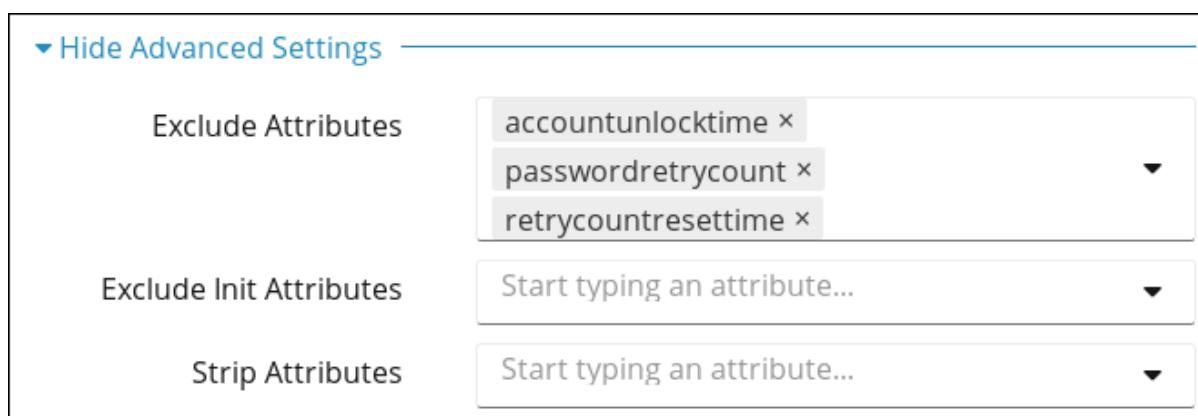
attributes from the replication agreement.

For details about configuring replication, see:

- [Section 15.3, “Single-supplier Replication”](#)
- [Section 15.4, “Multi-Supplier Replication”](#)
- [Section 15.5, “Cascading Replication”](#)

When you create the replication agreement in the procedures linked above, configure fractional replication:

1. When configuring the replication agreement on the supplier, click **Show Advanced Settings**.
2. Enter the ***passwordRetryCount***, ***retryCountResetTime***, and ***accountUnlockTime*** attributes names in to the **Exclude Attributes** field.



3. Finish configuring the replication agreement.

## 20.12. ENABLING DIFFERENT TYPES OF BINDS

Whenever an entity logs into or accesses the Directory Server, it *binds* to the directory. There are many different types of bind operation, sometimes depending on the method of binding (such as simple binds or autobind) and some depending on the identity of user binding to the directory (anonymous and unauthenticated binds).

The following sections contain configuration parameters that can increase the security of binds (as in [Section 20.12.1, “Requiring Secure Binds”](#)) or streamline bind operations (such as [Section 20.12.4, “Configuring Autobind”](#)).

### 20.12.1. Requiring Secure Binds

A simple bind is when an entity uses a simple bind DN–password combination to authenticate to the Directory Server. Although it is possible to use a password file rather than sending a password directly through the command line, both methods still require sending or accessing a plaintext password over the wire. That makes the password vulnerable to anyone sniffing the connection.

It is possible to require simple binds to occur over a secure connection (TLS or STARTTLS), which effectively encrypts the plaintext password as it is sent with the bind operation. (It is also possible to use alternatives to simple binds, such as SASL authentication and certificate-based authentication.)



## IMPORTANT

Along with regular users logging into the server and LDAP operations, server-to-server connections are affected by requiring secure connections for simple binds. Replication, synchronization, and database chaining can all use simple binds between servers, for instance.

Make sure that replication agreements, sync agreements, and chaining configuration specify secure connections if the ***nsslapd-require-secure-binds*** attribute is turned on. Otherwise, these operations will fail.



## NOTE

Requiring a secure connection for bind operations only applies to *authenticated binds*. Bind operations without a password (anonymous and unauthenticated binds) can proceed over standard connections.

1. Set the ***nsslapd-require-secure-binds*** configuration parameter to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-require-secure-binds=on
```

2. Restart the instance:

```
# dsctl instance_name restart
```

### 20.12.2. Disabling Anonymous Binds

If a user attempts to connect to the Directory Server without supplying any user name or password, this is an *anonymous bind*. Anonymous binds simplify common search and read operations, like checking the directory for a phone number or email address, by not requiring users to authenticate to the directory first.



## NOTE

By default, anonymous binds are allowed (on) for search and read operations. This allows access to *regular directory entries*, which includes user and group entries as well as configuration entries like the root DSE. A different option, ***rootdse***, allows anonymous search and read access to search the root DSE itself, but restricts access to all other directory entries.

However, there are risks with anonymous binds. Adequate ACIs must be in place to restrict access to sensitive information and to disallow actions like modifies and deletes. Additionally, anonymous binds can be used for denial of service attacks or for malicious people to gain access to the server.

[Section 18.10.1.1.3, “Granting Anonymous Access”](#) has an example on setting ACIs to control what anonymous users can access, and [Section 14.4.4, “Setting Resource Limits on Anonymous Binds”](#) has information on placing resource limits for anonymous users.

If those options do not offer a sufficient level of security, then anonymous binds can be disabled entirely:

1. Set the ***nsslapd-allow-anonymous-access*** configuration parameter to **off**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-allow-anonymous-access=off
```

2. Restart the instance:

```
# dsctl instance_name restart
```



#### NOTE

With anonymous binds disabled, the users cannot log in using their RDN. They are required to provide the full DN to log in.

In addition, when you disable anonymous binds, unauthenticated binds are also disabled automatically.

### 20.12.3. Allowing Unauthenticated Binds

Unauthenticated binds are connections to Directory Server where a user supplies an empty password. Using the default settings, Directory Server denies access in this scenario for security reasons:

```
# ldapsearch -w "" -p 389 -h server.example.com -b "dc=example,dc=com" \
-s sub -x "(objectclass=*)"
```

```
ldap_bind: Server is unwilling to perform (53)
additional info: Unauthenticated binds are not allowed
```



#### WARNING

Red Hat recommends not enabling unauthenticated binds. This authentication method enables users to bind without supplying a password as any account, including the Directory Manager. After the bind, the user can access all data with the permissions of the account used to bind.

To enable insecure unauthenticated binds, set the ***nsslapd-allow-unauthenticated-binds*** configuration option to **on**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-allow-unauthenticated-binds=on
```

### 20.12.4. Configuring Autobind

*Autobind* is a way to connect to the Directory Server based on local UNIX credentials, which are mapped to an identity stored in the directory itself. Autobind is configured in two parts:

Before configuring autobind, first make sure that LDAPI is enabled. Then, configure the autobind mappings (in [Section 20.12.4.2, “Configuring the Autobind Feature”](#)).

### 20.12.4.1. Overview of Autobind and LDAPI

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

The Directory Server uses these LDAPI connections to allow users to bind immediately to the Directory Server or to access the Directory Server using tools which support connections over Unix sockets. Autobind uses the *uid:gid* of the Unix user and maps that user to an entry in the Directory Server, then allows access for that user.

Autobind allows mappings to three directory entries:

- User entries, if the Unix user matches one user entry
- Directory Manager if the Unix user is **root** or the super user defined in ***nsslapd-ldapimapprootdn***

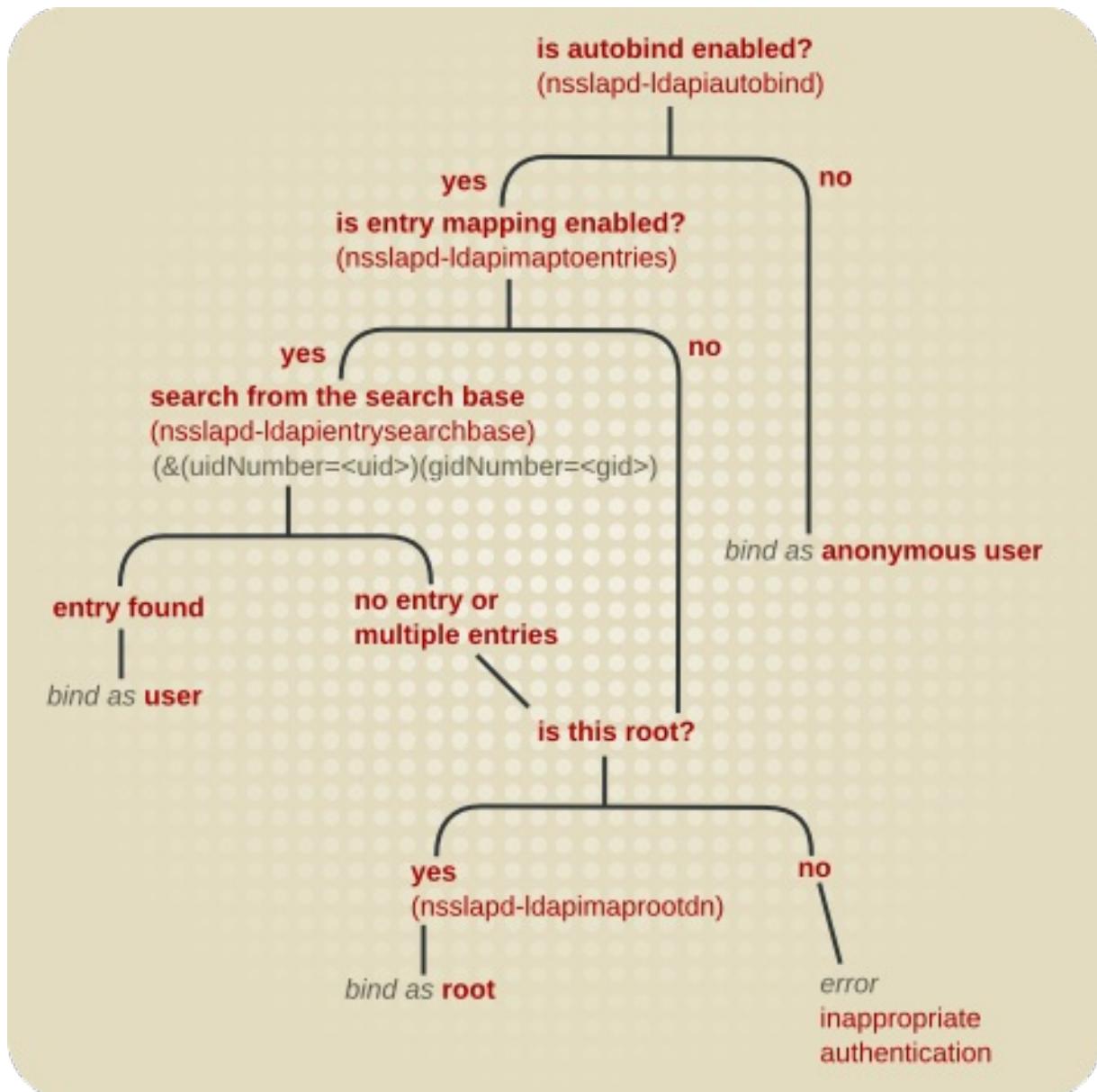


Figure 20.1. Autobind Connection Path

The special autobind users are entries beneath a special autobind suffix (outside the regular user subtree). The entries underneath are identified by their user and group ID numbers:

`gidNumber=gid+uidNumberuid, autobindsuffix`

If autobind is not enabled but LDAPI is, then Unix users are anonymously bound to the Directory Server, unless they provide other bind credentials.



### NOTE

Autobind allows a client to send a request to the Directory Server without supplying a bind user name and password or using other SASL authentication mechanism. According to the LDAP standard, if bind information is not given with the request, the server processes the request as an anonymous bind. To be compliant with the standard, which requires some kind of bind information, any client that uses autobind should send the request with SASL/EXTERNAL.

For more information on configuring SASL, see [Section 9.10, “Setting up SASL Identity Mapping”](#).

#### 20.12.4.2. Configuring the Autobind Feature

Enabling the **Autobind** feature allows only anonymous access to Directory Server. However, you can configure to map Linux users to Directory Server entries and also to map the **root** user to the Directory Manager:

1. Verify that the **nsslapd-ldapiautobind** parameter is enabled, which is the default:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-ldapiautobind
nsslapd-ldapiautobind: on
```

2. If **nsslapd-ldapiautobind** parameter is set to **off**, enable it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-ldapiautobind=on
```

3. To map user entries, set, for example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-ldapimaptorents=on nsslapd-ldapuidnumbertype=uidNumber nsslapd-ldapgidnumbertype=gidNumber nsslapd-ldapientrysearchbase=ou=People,dc=example,dc=com
```

- **nsslapd-ldapimaptorents=on** enables entry mapping.
- **nsslapd-ldapuidnumbertype=uidNumber** sets the attribute in Directory Server that contains the Unix UID number.
- **nsslapd-ldapgidnumbertype=gidNumber** sets the attribute in Directory Server that contains the Unix GID number.
- **nsslapd-ldapientrysearchbase=ou=People,dc=example,dc=com** sets the DN where to search user entries.

4. Optionally, to map the **root** user in Red Hat Enterprise Linux to the **cn=Directory Manager** account in Directory Server:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-ldapimaprootdn="cn=Directory Manager"
```

5. Restart the instance:

```
# dsctl instance_name restart
```

## 20.13. USING PASS-THROUGH AUTHENTICATION

Pass-through authentication (PTA) is a mechanism which allows one Red Hat Directory Server instance to consult another to authenticate bind requests. Pass-through authentication is implemented through the PTA Plug-in; when enabled, the plug-in lets a Directory Server instance accept simple bind operations (password-based) for entries not stored in its local database.

Directory Server uses PTA to administer the user and configuration directories on separate instances of Directory Server.

The first instance acts as the PTA Directory Server which is the server that passes through bind requests to another Directory Server. The second instance acts as the authenticating directory, which is the server that contains the entry and verifies the bind credentials of the requesting client.

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.

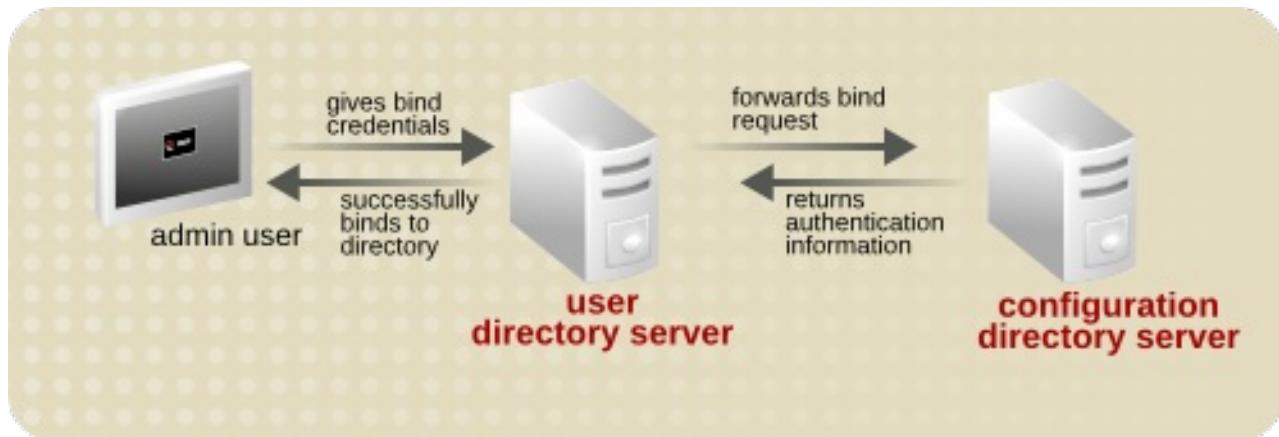


Figure 20.2. Simple Pass-Through Authentication Process

Here's how pass-through authentication works:

1. The configuration Directory Server (authenticating directory) is installed on machine A. The configuration directory always contains the suffix with the authenticating user entry, for example, **o=RedHat**. In this example, the server name is **authdir.example.com**.
2. The user Directory Server (PTA directory) is then installed on machine B. The user directory stores the root suffix, such as **dc=example,dc=com**. In this example, the server name is **userdir.example.com**.
3. Set up the plug-in on **userdir.example.com** by using the following commands:

```
# dsconf -D "cn=Directory Manager" ldap://userdir.example.com plugin pass-through-auth
enable
# dsconf -D "cn=Directory Manager" ldap://userdir.example.com plugin pass-through-auth url
add "ldap://authdir.example.com/o=RedHat"
```

4. Restart Directory Server on **userdir.example.com**.
5. The user directory is now configured to send all bind requests for entries with a DN containing **o=RedHat** to the configuration directory **authdir.example.com**.
6. The user directory allows any user from **o=RedHat** to bind.

### 20.13.1. PTA Plug-in Syntax

PTA Plug-in configuration information is specified in the **cn=Pass Through Authentication, cn=plugins,cn=config** entry on the PTA directory (the user directory configured to pass through bind requests to the authenticating directory) using the required PTA syntax.

Use the following commands to manage pass-through authentication URLs:

- To add a pass-through authentication URL:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth url
add URL
```

- To modify a pass-through authentication URL:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth url
modify old_URL new_URL
```

- To remove pass-through authentication URL:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth url
delete URL
```

The variable components of the PTA plug-in syntax are described in [Table 20.3, “PTA Plug-in Parameters”](#).



#### NOTE

The LDAP URL (**ldap|ldaps://authDS/subtree**) must be separated from the optional parameters (*maxconns*, *maxops*, *timeout*, *ldver*, *connlifetime*, *startTLS*) by a single space. If any of the optional parameters are defined, all of them must be defined, even if only the default values are used.

Several authenticating directories or subtrees can be specified by incrementing the **nsslapd-pluginarg** attribute suffix by one each time, as in [Section 20.13.3.2, “Specifying Multiple Authenticating Directory Servers”](#). For example:

```
nsslapd-pluginarg0: LDAP URL for the first server
nsslapd-pluginarg1: LDAP URL for the second server
nsslapd-pluginarg2: LDAP URL for the third server
...
```

The optional parameters are described in the following table in the order in which they appear in the syntax.

**Table 20.3. PTA Plug-in Parameters**

Variable	Definition		
state	Defines whether the plug-in is enabled or disabled. Acceptable values are <b>on</b> or <b>off</b> .		
ldap ldaps	Defines whether TLS is used for communication between the two Directory Servers. See <a href="#">Section 20.13.2.1, "Configuring the Servers to Use a Secure Connection"</a> for more information.		
authDS	<p>The authenticating directory host name. The port number of the Directory Server can be given by adding a colon and then the port number. For example, <b>ldap://dirserver.example.com:389/</b>. If the port number is not specified, the PTA server attempts to connect using either of the standard ports:</p> <table border="1" style="margin-left: 20px;"> <tr> <td>Port 389 if <b>ldap://</b> is specified in the URL.</td></tr> <tr> <td>Port 636 if <b>ldaps://</b> is specified in the URL.</td></tr> </table> <p>See <a href="#">Section 20.13.2.2, "Specifying the Authenticating Directory Server"</a> for more information.</p>	Port 389 if <b>ldap://</b> is specified in the URL.	Port 636 if <b>ldaps://</b> is specified in the URL.
Port 389 if <b>ldap://</b> is specified in the URL.			
Port 636 if <b>ldaps://</b> is specified in the URL.			
subtree	The <i>pass-through subtree</i> . The PTA Directory Server passes through bind requests to the authenticating Directory Server from all clients whose DN is in this subtree. See <a href="#">Section 20.13.2.3, "Specifying the Pass-Through Subtree"</a> for more information. This subtree must not exist on this server. To pass the bind requests for <b>o=NetscapeRoot</b> to the configuration directory, the subtree <b>o=NetscapeRoot</b> must not exist on the server.		
maxconns	<i>Optional</i> . The maximum number of connections the PTA directory can simultaneously open to the authenticating directory. The default is <b>3</b> . See <a href="#">Section 20.13.2.4, "Configuring the Optional Parameters"</a> for more information.		
maxops	<i>Optional</i> . 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 <b>5</b> . See <a href="#">Section 20.13.2.4, "Configuring the Optional Parameters"</a> for more information.		
timeout	<i>Optional</i> . 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 <b>300</b> seconds (five minutes). Specify zero ( <b>0</b> ) to indicate no time limit should be enforced. See <a href="#">Section 20.13.2.4, "Configuring the Optional Parameters"</a> for more information.		
ldver	<i>Optional</i> . 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, and Red Hat strongly recommends <i>against</i> using LDAPv2, which is old and will be deprecated. See <a href="#">Section 20.13.2.4, "Configuring the Optional Parameters"</a> for more information.		

Variable	Definition
connlifetime	<i>Optional.</i> 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 this option is not specified, or if only one host is listed, no connection lifetime will be enforced. If two or more hosts are listed, the default is <b>300</b> seconds (five minutes). See <a href="#">Section 20.13.2.4, “Configuring the Optional Parameters”</a> for more information.
startTLS	<i>Optional.</i> A flag of whether to use STARTTLS for the connection to the authenticating directory. STARTTLS establishes a secure connection over the standard port, so it is useful for connecting using LDAP instead of LDAPS. The TLS server and CA certificates need to be available on both of the servers.  The default is <b>0</b> , which is off. To enable STARTTLS, set it to <b>1</b> . To use STARTTLS, the LDAP URL must use <b>ldap:</b> , not <b>ldaps:</b>  See <a href="#">Section 20.13.2.4, “Configuring the Optional Parameters”</a> for more information.

## 20.13.2. Configuring the PTA Plug-in

To modify the PTA configuration:

1. Use the **dsconf plugin pass-through-auth** command to modify the **cn=Pass Through Authentication,cn=plugins,cn=config** entry.
2. Restart Directory Server.

Before configuring any of the PTA Plug-in parameters, the PTA Plug-in entry must be present in the Directory Server. If this entry does not exist, create it with the appropriate syntax, as described in [Section 20.13.1, “PTA Plug-in Syntax”](#).



### NOTE

If the user and configuration directories are installed on different instances of the directory, the PTA Plug-in entry is automatically added to the user directory's configuration and enabled.

This section provides information about configuring the plug-in in the following sections:

- [Section 20.13.2.1, “Configuring the Servers to Use a Secure Connection”](#)
- [Section 20.13.2.2, “Specifying the Authenticating Directory Server”](#)
- [Section 20.13.2.3, “Specifying the Pass-Through Subtree”](#)
- [Section 20.13.2.4, “Configuring the Optional Parameters”](#)

### 20.13.2.1. Configuring the Servers to Use a Secure Connection

The PTA directory can be configured to communicate with the authenticating directory over TLS by specifying LDAPS in the LDAP URL of the PTA directory. For example:

```
nsslapd-pluginarg0: ldaps://ldap.example.com:636/o=NetscapeRoot
```

### 20.13.2.2. 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 defined as the authenticating directory. To specify the authenticating Directory Server, replace *authDS* in the LDAP URL of the PTA directory with the authenticating directory's host name, as described in [Table 20.3, “PTA Plug-in Parameters”](#).

1. Use the **dsconf plugin pass-through-auth** command to edit the PTA Plug-in entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth add
ldap://server.example.com/o=example
```

Optionally, include the port number. If the port number is not given, the PTA Directory Server attempts to connect using either the standard port (389) for **ldap://** or the secure port (636) for **ldaps://**.

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. There can be multiple authenticating Directory Servers specified, as required, to provide failover if the first Directory Server is unavailable. All of the authentication Directory Server are set in the **nsslapd-pluginarg0** attribute.

Multiple authenticating Directory Servers are listed in a space-separate list of *host:port* pairs, with this format:

```
ldap|ldaps://host1:port1 host2:port2/subtree
```

2. Restart the server.

```
# dsctl instance_name restart
```

### 20.13.2.3. Specifying the Pass-Through Subtree

The PTA directory passes through bind requests to the authenticating directory from all clients with a DN defined in the pass-through subtree. The subtree is specified 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.

1. Use the **dsconf plugin pass-through-auth** command to import the LDIF file into the directory:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth add
ldap://server.example.com/o=example
```

For information on the variable components in this syntax, see [Table 20.3, “PTA Plug-in Parameters”](#).

2. Restart the server:

```
# dsctl instance_name restart
```

#### 20.13.2.4. Configuring the Optional Parameters

Additional parameters the control the PTA connection can be set with the LDAP URL.

```
ldap|ldaps://authDS/subtree maxconns, maxops, timeout, ldver, connlifetime, startTLS
```

- 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 *maxops*. The default is value is **5**.
- The time limit for the PTA Directory Server to wait for a response from the authenticating Directory Server. In the PTA syntax, this parameter is *timeout*. The default value is **300** seconds (five minutes).
- The version of the LDAP protocol for the PTA Directory Server to use to connect to the authenticating Directory Server. In the PTA syntax, this parameter is *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 this option is not specified 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 *connlifetime*.
- Whether to use STARTTLS for the connection. STARTTLS creates a secure connection over a standard LDAP port. For STARTTLS, the servers must have their server and CA certificates installed, but they do not need to be running in TLS.

The default is **0**, which means STARTTLS is off. To enable STARTTLS, set it to **1**. To use STARTTLS, the LDAP URL must use **ldap:**; not **ldaps:**.

1. Use the **dsconf plugin pass-through-auth** command to edit the plug-in entry:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth add
ldap://server.example.com/o=example 3,5,300,3,300,0
```

(In this example, each of the optional parameters is set to its default value.) Make sure there is a space between the *subtree* parameter, and the optional parameters.



#### NOTE

Although these parameters are optional, if any one of them is defined, they all must be defined, even if they use the default values.

2. Restart the server:

```
# dsctl instance_name restart
```

### 20.13.3. PTA Plug-in Syntax Examples

This section contains the following examples of PTA Plug-in syntax in the **dse.ldif** file:

- [Section 20.13.3.1, "Specifying One Authenticating Directory Server and One Subtree"](#)
- [Section 20.13.3.2, "Specifying Multiple Authenticating Directory Servers"](#)
- [Section 20.13.3.3, "Specifying One Authenticating Directory Server and Multiple Subtrees"](#)
- [Section 20.13.3.4, "Using Non-Default Parameter Values"](#)
- [Section 20.13.3.5, "Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers"](#)

#### 20.13.3.1. 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 host name of the authenticating Directory Server is **configdir.example.com**.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
...
```

#### 20.13.3.2. 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. There can be multiple authenticating Directory Servers specified, as required, to provide failover if the first Directory Server is unavailable. All of the authentication Directory Server are set in the **nsslapd-pluginarg0** attribute. Multiple authenticating Directory Servers are listed in a space-separated list of *host:port* pairs. For example:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com:389 config2dir.example.com:1389/o=NetscapeRoot
...
```



#### NOTE

The **nsslapd-pluginarg0** attribute sets the authentication Directory Server; additional **nsslapd-pluginargN** attributes can set additional *suffixes* for the PTA Plug-in to use, but not additional *hosts*.

#### 20.13.3.3. 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
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
nsslapd-pluginarg1: ldap://configdir.example.com/dc=example,dc=com
...
```

#### 20.13.3.4. 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
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot 10,5,300,3,300,1
...
```

#### 20.13.3.5. Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

To specify a different pass-through subtree and optional parameter values for each authenticating Directory Server, set 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
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0:ldap://configdir.example.com/o=NetscapeRoot 10,15,30,3,600,0
nsslapd-pluginarg1:ldap://config2dir.example.com/dc=example,dc=com 7,7,300,3,300,1
...
```

## 20.14. USING ACTIVE DIRECTORY-FORMATTED USER NAMES FOR AUTHENTICATION

When you connect to Directory Server, you must specify the distinguished name (DN) of the user, such as **uid=user\_name,ou=People,dc=example,dc=com**, to authenticate. However, the DN can be difficult to remember. If you enable and configure the **AD DN** plug-in, you can use Active Directory-formatted user names, such as **user\_name** or **user\_name@domain** instead of the DN.

After you enable the plug-in and a user connects to the directory using a user name that is not DN-formatted, Directory Server searches the DN based on the plug-in's configuration. If the search returns one DN, Directory Server uses this DN for the authentication. If none or multiple DNs are returned, authentication fails.



#### NOTE

You can only enable and configure the **AD DN** plug-in using the command line.

To enable and configure the plug-in it to use **example.com** as the default domain:

1. Add the **cn=addn,cn=plugins,cn=config** plug-in entry and set the default domain:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=addn,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: addn
nsslapd-pluginPath: libaddn-plugin
nsslapd-pluginInitfunc: addn_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginId: addn
nsslapd-pluginVendor: 389 Project
nsslapd-pluginVersion: 1.3.6.0
nsslapd-pluginDescription: Allow AD DN style bind names to LDAP
addn_default_domain: example.com
```

The required ***addn\_default\_domain*** parameter in the plug-in entry sets the default domain. The plug-in appends this domain if the specified user name during an authentication does not contain a domain name.

2. Add a configuration entry for the default domain:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example.com,cn=addn,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
cn: example.com
addn_base: ou=People,dc=example,dc=com
addn_filter: (&(objectClass=account)(uid=%s))
```

For details about the parameters used in the example, see their descriptions in the [Red Hat Directory Server Configuration, Command, and File Reference](#).



### WARNING

You must add at least a configuration entry for the default domain. If the entry is missing, Directory Server fails to start.

3. Optionally, you can create additional domain configurations as described in the previous step to support multiple domain names. Each domain configuration can use a different search base and filter.
4. Restart the Directory Server instance:

```
# dsctl instance_name restart
```

## 20.15. USING PAM FOR PASS-THROUGH AUTHENTICATION

Many systems already have authentication mechanisms in place for Unix and Linux users. One of the most common authentication frameworks is *Pluggable Authentication Modules* (PAM). Since many networks already have existing authentication services available, administrators may want to continue using those services. A PAM module can be configured to tell Directory Server to use an existing authentication store for LDAP clients.

PAM pass-through authentication in Red Hat Directory Server uses the PAM Pass-Through Authentication Plug-in, which enables the Directory Server to talk to the PAM service to authenticate LDAP clients.

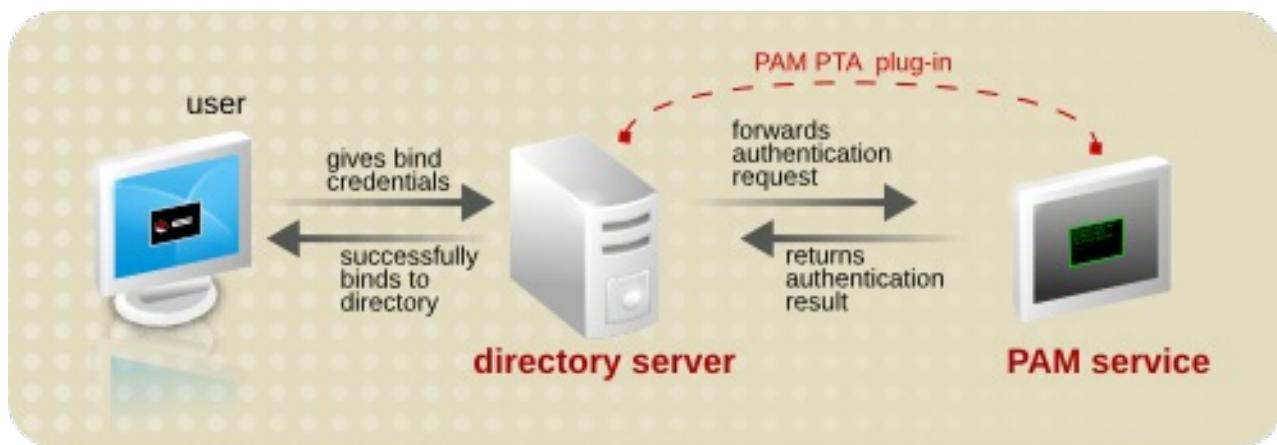


Figure 20.3. PAM Pass-Through Authentication Process



### NOTE

PAM pass-through authentication works together with account inactivation when authenticating users, assuming that the appropriate mapping method (ENTRY) is used. However, PAM pass-through authentication does not validate passwords against password policies set either globally or locally, because the passwords are set and stored in the PAM module, not in the Directory Server.

### 20.15.1. PAM Pass-Through Authentication Configuration Options

PAM pass-through authentication is configured in child entries beneath the PAM Pass-Through Authentication plug-in container entry. There can be multiple PAM pass-through authentication policies, applied to different suffixes or to different entries within suffixes.

There are several different areas that can be configured for PAM pass-through:

- The suffixes that are controlled by the PAM pass-through authentication plug-in. This covers suffixes to exclude, suffixes to include, and how to handle a missing suffix.
- Individual entries within the configured suffixes which are the target of the authentication configuration. By default, all entries within a suffix are included in the authentication scope, but it is possible to configure multiple, different PAM Pass-Through Auth plug-in instances and then apply different plug-in configuration to different users.

- The PAM attribute mapping. The credentials that are offered to the Directory Server have to be mapped in some way to an LDAP entry and then, back to the credentials in the PAM service. This is done by defining a mapping method and then, optionally, which LDAP attribute to use to match the credentials.
- General configuration such as using TLS connections, the PAM service to use, and whether to fallback to LDAP authentication if PAM authentication fails.



#### NOTE

There can be multiple configuration instances of the PAM Pass-Through Authentication plug-in. An instance of the PAM Pass-Through Authentication plug-in can be applied to a subset of user entries by using the **pamFilter** attribute to set an LDAP filter to search for the specific entries to use with the plug-in.

For a list of attributes you can set, see the [PAM Pass Through Auth Plug-in Attributes](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

#### 20.15.1.1. Specifying the Suffixes to Target for PAM PTA

The PAM PTA plug-in is applied globally, to all suffixes, by default unless they are explicitly excluded. Excluding and including suffixes can help target what areas in the directory use PAM authentication instead of LDAP authentication.



#### NOTE

The target of a PAM pass-through authentication entry must be a suffix, not an arbitrary subtree. As described in [Section 2.1, “Creating and Maintaining Suffixes”](#), a suffix is a subtree which is associated with a specific back end database, such as **cn=config** which is associated with the root suffix **dc=example,dc=com** which is associated with **userRoot**.

The **pamExcludeSuffix** attribute excludes a suffix. By default, only the configuration subtree (**cn=config**) is excluded. Alternatively, the PAM PTA plug-in can be applied to a suffix with the **pamIncludeSuffix** attribute. Both of these attributes are multi-valued.

If the include attribute is set, for example, all other suffixes are automatically excluded. Likewise, if an exclude attribute is set, all other suffixes are automatically included.

```
pamExcludeSuffix: cn=config
```

With **pamIncludeSuffix**, only the given suffix is included and all others are automatically excluded. Since this attribute is multi-valued, more than one suffix can be included in the PAM evaluation by explicitly listing the suffixes.

```
pamIncludeSuffix: ou=Engineering,dc=example,dc=com  
pamIncludeSuffix: ou=QE,dc=example,dc=com
```

The **pamMissingSuffix** attribute tells the server how to handle a failure if the specified suffix (include or exclude) does not exist. If it is set to **IGNORE**, then if the suffix does not exist, the plug-in simply skips that suffix and tries the next.

```
pamMissingSuffix: IGNORE
pamIncludeSuffix: ou=Engineering,dc=example,dc=com
pamIncludeSuffix: ou=Not Real,dc=example,dc=com
```

### 20.15.1.2. Applying Different PAM Pass-Through Authentication Configurations to Different Entries

By default, a PAM pass-through authentication policy applies to all entries within the designated suffixes. However, it is possible to specify an LDAP filter in the **pamFilter** attribute which identifies specific entries within the suffix to which to apply the PAM pass-through authentication policy.

This is useful for applying different PAM configurations or mapping methods to different user types, using multiple PAM pass-through authentication policies.

### 20.15.1.3. Setting PAM PTA Mappings

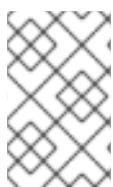
There has to be a way to connect the LDAP identity to the PAM identity. The first thing to define is the *method* to use to map the entries. There are three options: DN, RDN, and ENTRY. ENTRY uses a user-defined attribute in the entry.

Multiple mapping methods can be supplied in an ordered, space-separated list. The plug-in attempts to use each mapping method in the order listed until authentication succeeds or until it reaches the end of the list.

For example, this mapping method first maps the RDN method, then ENTRY, then DN, in the order the methods are listed:

```
pamIDMapMethod: RDN ENTRY DN
```

The different mapping methods are listed in [Table 20.4, “Mapping Methods for PAM Authentication”](#).



#### NOTE

Directory Server user account inactivation is only validated using the ENTRY mapping method. With RDN or DN, a Directory Server user whose account is inactivated can still bind to the server successfully.

**Table 20.4. Mapping Methods for PAM Authentication**

Mapping	Description
RDN	This method uses the value from the leftmost RDN in the bind DN. The mapping for this method is defined by Directory Server. This is the default mapping method, if none is given.
ENTRY	This method pulls the value of the PAM identity from a user-defined attribute in the bind DN entry. The identity attribute is defined in the <b>pamIDAttr</b> attribute. For example: <b>pamIDAttr: customPamUid</b>
DN	This method uses the full distinguished name from the bind DN. The mapping for this method is defined by Directory Server.

### 20.15.1.4. Configuring General PAM PTA Settings

Three general configuration settings can be set for PAM authentication:

- The service name to send to PAM (**pamService**); this is the name of the configuration file to use in **/etc/pam.d**
- Whether to require a secure connection (**pamSecure**)
- Whether to fall back to LDAP authentication if PAM authentication fails (**pamFallback**)

```
pamFallback: false
pamSecure: false
pamService: ldapserver
```

### 20.15.2. Configuring PAM Pass-Through Authentication



#### NOTE

There can be multiple configuration instances of the PAM Pass-Through Authentication plug-in. An instance of the PAM Pass-Through Authentication plug-in can be applied to a subset of user entries by using the **pamFilter** attribute to set an LDAP filter to search for the specific entries to use with the plug-in.

PAM pass-through authentication is configured through the command line.

1. Make sure the PAM service is fully configured.
2. Remove the **pam\_fprintd.so** module from the PAM configuration file.



#### IMPORTANT

The **pam\_fprintd.so** module cannot be in the configuration file referenced by the **pamService** attribute of the PAM Pass-Through Authentication Plug-in configuration. Using the PAM **fprintd** module causes the Directory Server to hit the max file descriptor limit and can cause the Directory Server process to abort.

3. Enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin set "PAM Pass-Through Auth Plugin" --enabled on
```

4. Create the PAM Pass-Through Auth plug-in configuration entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth
pam-config "Admin PAM PTA Config" add --exclude-suffix="cn=config" --
id_map_method="RDN ENTRY" --id-attr="customPamUid" --
filter="(manager=uid=example_user,ou=people,dc=example,dc=com pamFallback: FALSE" -
-secure="TRUE" --service="ldapserver"
```

5. Restart the instance:

```
# dsctl instance_name restart
```

### 20.15.3. Using PAM Pass-Through Authentication with Active Directory as the Back End

PAM pass-through authentication forwards the credentials from the Directory Server to the PAM service. One option is to set up and configure PAM modules specifically for Directory Server. Another option – and one which may be more repeatable and more convenient in some infrastructures – is to use the System Security Services Daemon (SSSD) to configure PAM. Because SSSD can use a variety of different identity stores, a lot of different servers or services can be used to provide credentials, including Active Directory.

Using pass-through authentication through SSSD is a daisy chain of services. The PAM PTA Plug-in is configured as normal. It points to the given PAM service file to use. This service file is managed by SSSD, and SSSD is configured to connect with whatever identity provider is required, even multiple providers.

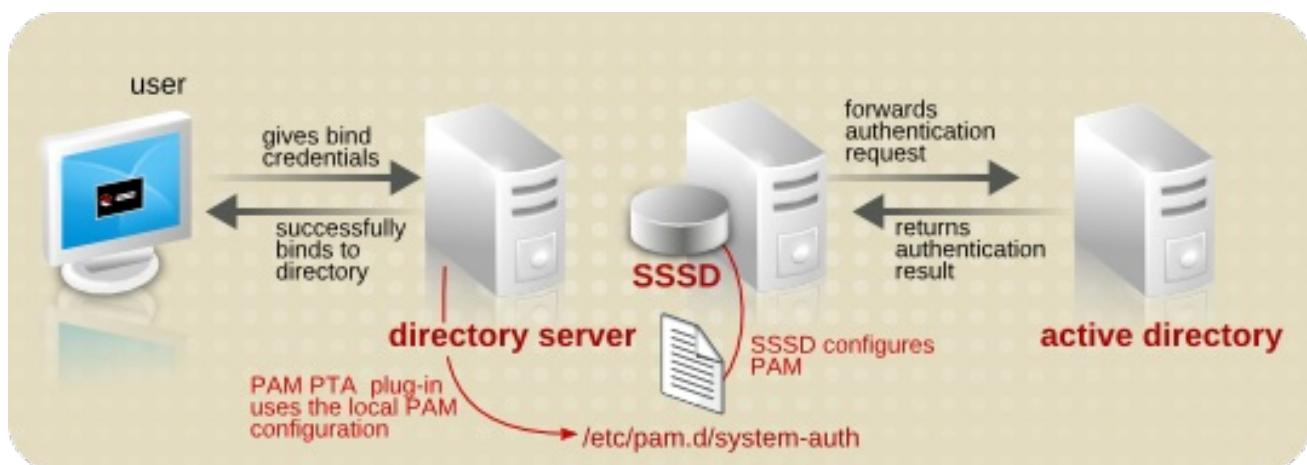


Figure 20.4. PAM Pass-Through Authentication with SSSD

To configure PAM pass-through authentication with Active Directory:

- Configure SSSD to use the Active Directory server as one of its identity providers.

This configuration is covered in the [Using Active Directory as an Identity Provider for SSSD](#) section in the *Windows Integration Guide*.

- Enable the PAM Pass-Through Auth plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin set "PAM Pass-Through Auth Plugin" --enabled on
```

- Create the PAM Pass-Through Auth plug-in configuration entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin pass-through-auth pam-config "AD PAM PTA Config" add --id_map_method="ENTRY" --id-attr="sAMAccountName" --service="system-auth"
```

- Restart the server to load the plug-in configuration.

```
# dsctl instance_name restart
```

## 20.16. MANUALLY INACTIVATING USERS AND ROLES

A single user account or set of accounts can be temporarily inactivated. Once an account is 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.

The same procedures are used to deactivate users and roles. However, when a role is deactivated, the *members of the role* are deactivated, not the role entry itself. For more information about roles in general and how roles interact with access control in particular, see [Chapter 8, Organizing and Grouping Entries](#).



### WARNING

The root entry (the entry corresponding to the root or sub suffix) on a database cannot be deactivated. [Chapter 3, Managing Directory Entries](#) has information on creating the entry for a root or sub suffix, and [Chapter 2, Configuring Directory Databases](#) has information on creating root and sub suffixes.

### 20.16.1. Displaying the Status of an Account or Role

To display the status of:

- An account, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account entry-status "uid=user_name,ou=People,dc=example,dc=com"
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20200813085535Z (2020-08-13 08:55:35)
Entry Modification Date: 20200813085535Z (2020-08-13 08:55:35)
Entry State: activated
```

Optional: Pass the **-V** option to the command to display additional details:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account entry-status "uid=user_name,ou=People,dc=example,dc=com" -V
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20200824160645Z (2020-08-24 16:06:45)
Entry Modification Date: 20200824160645Z (2020-08-24 16:06:45)
Entry Last Login Date: 20200824160645Z (2020-08-24 16:06:45)
Entry Time Until Inactive: 2 seconds (2020-08-24 16:07:45)
Entry State: activated
```

The previous output is an example of an active account, as indicated by the last two lines of the output. An inactive account would instead provide output similar to the following:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account entry-status "uid=user_name,ou=People,dc=example,dc=com" -V
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20200824160645Z (2020-08-24 16:06:45)
```

```
Entry Modification Date: 20200824160645Z (2020-08-24 16:06:45)
Entry Last Login Date: 20200824160645Z (2020-08-24 16:06:45)
Entry Time Since Inactive: 3 seconds (2020-08-24 16:07:45)
Entry State: inactivity limit exceeded
```

- A role, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" role
entry-status "cn=Marketing,ou=People,dc=example,dc=com"
Entry DN: cn=Marketing,ou=people,dc=example,dc=com
Entry State: activated
```

To display the status of a sub-tree instead of an entry, use the **subtree-status** instead of the **entry-status** option. When you use the **subtree-status** option, you can specify a filter (**-f**) and a search scope (**-s**) to narrow down the results. Additionally, you can refine the search using the **-i** option to return only inactive accounts or the **-o date** option to return only accounts which will be inactive before the specified date:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" account
account "ou=People,dc=example,dc=com" -f "(uid=*)" -V -o "2020-08-25T14:30:30"
```

Specify the date in the following format: YYYY-MM-DDTHH:MM:SS

## 20.16.2. Inactivating and Activating Users and Roles Using the Command Line

To inactivate:

- A user account, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account lock "uid=user_name,ou=People,dc=example,dc=com"
```

- A role, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" role
lock "cn=Marketing,ou=People,dc=example,dc=com"
```

To activate:

- A user account, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com"
account unlock "uid=user_name,ou=People,dc=example,dc=com"
```

- A role, enter:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b "dc=example,dc=com" role
unlock "cn=Marketing,ou=People,dc=example,dc=com"
```

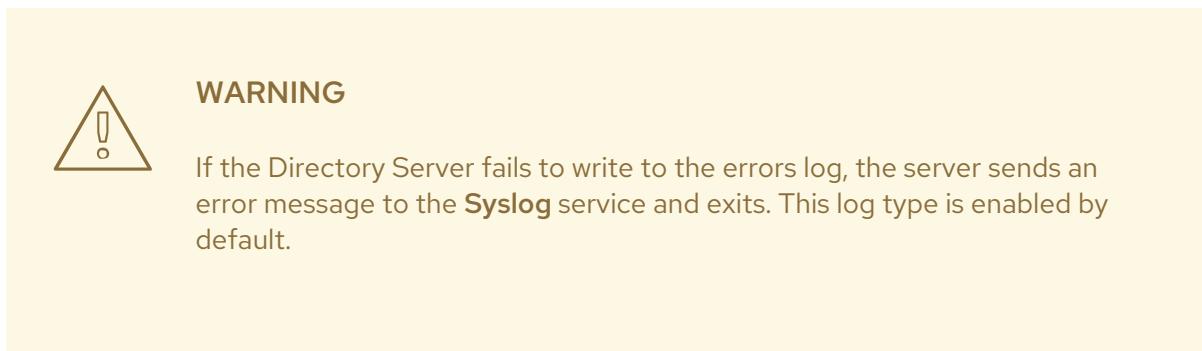
# CHAPTER 21. MONITORING SERVER AND DATABASE ACTIVITY

This chapter describes monitoring database and Red Hat Directory Server logs. For information on using SNMP to monitor the Directory Server, see [Section 21.10, “Monitoring Directory Server Using SNMP”](#).

## 21.1. TYPES OF DIRECTORY SERVER LOG FILES

Directory Server provides the following log types:

- Access log: Contains information on client connections and connection attempts to the Directory Server instance. This log type is enabled by default.
- Error log: Contains detailed messages of errors and events the directory experiences during normal operations. This log type is enabled by default.



- Audit log: Records changes made to each database as well as to server configuration. This log is not enabled by default.
- Audit fail log: Records failed audit events. This log is not enabled by default.

## 21.2. DISPLAYING LOG FILES

You can display the Directory Server log files using the command line and web console:

### 21.2.1. Displaying Log Files Using the Command Line

To display the log files using the command line, use the utilities included in Red Hat Enterprise Linux, such as **less**, **more**, and **cat**. For example:

```
# less /var/log/dirsrv/slapd-instance_name/access
```

To display the locations of log files:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-accesslog  
nsslapd-errorlog nsslapd-auditlog nsslapd-auditfaillog  
  
nsslapd-accesslog: /var/log/dirsrv/slapd-instance_name/access  
nsslapd-errorlog: /var/log/dirsrv/slapd-instance_name/errors  
nsslapd-auditlog: /var/log/dirsrv/slapd-instance_name/audit  
nsslapd-auditfaillog: /var/log/dirsrv/slapd-instance_name/audit-failure
```

**NOTE**

If logging for a log type is not enabled, the corresponding log file does not exist.

### 21.2.2. Displaying Log Files Using the Web Console

To display the Directory Server log files:

1. Open the Directory Server user interface in the web console. See [Section 1.4, "Logging Into Directory Server Using the Web Console"](#).
2. Select the instance.
3. Open the **Monitoring** menu.
4. Open the **Logging** menu, and select the log file you want to display.

5. Optionally, you can apply the following settings to the log file viewer:
  - Set the number of lines to display in the **Log Lines To Show** field.
  - Enable automatically displaying new log entries by selecting **Continuously Refresh**.
6. Click the **Refresh** button to apply the changes.

## 21.3. CONFIGURING LOG FILES

For all types of log files, the log *creation* and log *deletion* policies have to be configured. The log creation policy sets when a new log file is started, and the log deletion policy sets when an old log file is deleted.

### 21.3.1. Enabling or Disabling Logs

The access and error logging is enabled by default. However, audit and audit fail logging is disabled by default.

**NOTE**

Disabling the access logging can be useful in certain scenarios, because every 2000 accesses to the directory increases the log file by approximately 1 megabyte. However, before turning off access logging, consider that this information can help troubleshooting problems.

#### 21.3.1.1. Enabling or Disabling Logging Using the Command Line

Use the **dsconf config replace** command to modify the parameters in the **cn=config** subtree that control the Directory Server logging feature:

- Access log: **nsslapd-accesslog-logging-enabled**
- Error log: **nsslapd-errorlog-logging-enabled**
- Audit log: **nsslapd-auditlog-logging-enabled**
- Audit fail log: **nsslapd-auditfaillog-logging-enabled**

For further details, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

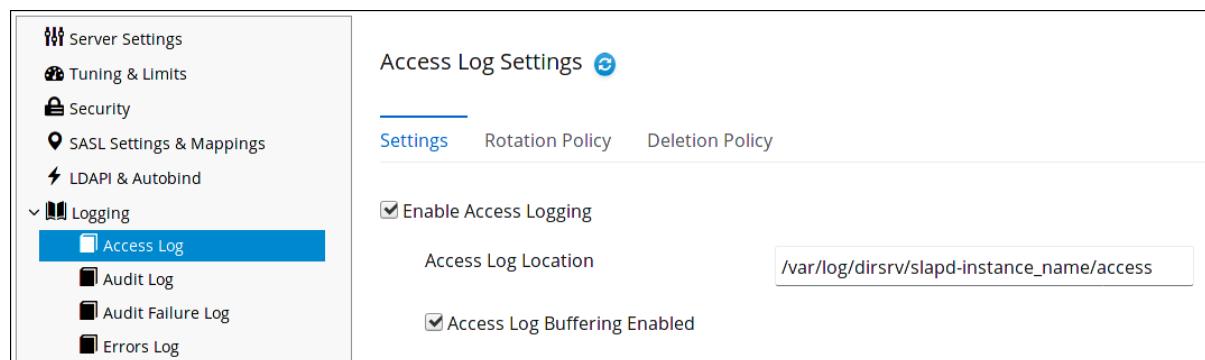
For example, to enable audit logging, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-auditlog-logging-enabled=on
```

### 21.3.1.2. Enabling or Disabling Logging Using the Web Console

To enable or disable logging in web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. Open the **Server Settings** menu, and select the log type you want to configure under the **Logging** entry.



4. Enable or disable the logging feature for the selected log type.
5. Optionally, set additional parameters to define, for example, a log rotation or log deletion policy.
6. Click **Save**.

### 21.3.2. Configuring Plug-in-specific Logging

For debugging, you can enable access and audit logging for operations a plug-ins executes. For details, see the **nsslapd-logAccess** and **nsslapd-logAudit** parameter in the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

### 21.3.3. Disabling High-resolution Log Time Stamps

Using the default settings, Directory Server logs entries with nanosecond precision:

```
[27/May/2016:17:52:04.754335904 -0500] schemareload - Schema validation passed.  
[27/May/2016:17:52:04.894255328 -0500] schemareload - Schema reload task finished.
```

To disable high-resolution log time stamps:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-logging-hr-timestamps-enabled=off
```



#### NOTE

The option to disable high-resolution log time stamps is deprecated and will be removed in a future release.

After disabling high-resolution log time stamps, Directory Server logs with second precision only:

```
[27/May/2016:17:52:04 -0500] schemareload - Schema validation passed.  
[27/May/2016:17:52:04 -0500] schemareload - Schema reload task finished.
```

### 21.3.4. Defining a Log File Rotation Policy

To periodically archive the current log file and create a new one, set a log file rotation policy. You can update the settings in the **cn=config** subtree using the command line or the web console.

You can set the following configuration parameters to control the log file rotation policy:

#### Access mode

The access mode sets the file permissions on newly created log files.

- Access log: ***nsslapd-accesslog-mode***
- Error log: ***nsslapd-errorlog-mode***
- Audit log: ***nsslapd-auditlog-mode***
- Audit fail log: ***nsslapd-auditfaillog-mode***

#### Maximum number of logs

Sets the maximum number of log files to keep. When the number of files is reached, Directory Server deletes the oldest log file before creating the new one.

- Access log: ***nsslapd-accesslog-maxlogsperdir***
- Error log: ***nsslapd-errorlog-maxlogsperdir***
- Audit log: ***nsslapd-auditlog-maxlogsperdir***
- Audit fail log: ***nsslapd-auditfaillog-maxlogsperdir***

#### File size for each log

Sets the maximum size of a log file in megabytes before it is rotated.

- Access log: ***nsslapd-accesslog-maxlogsize***
- Error log: ***nsslapd-errorlog-maxlogsize***
- Audit log: ***nsslapd-auditlog-maxlogsize***
- Audit fail log: ***nsslapd-auditfaillog-maxlogsize***

#### Create a log every

Sets the maximum age of a log file.

- ***nsslapd-accesslog-logrotationtime*** and ***nsslapd-accesslog-logrotationtimeunit***
- ***nsslapd-errorlog-logrotationtime*** and ***nsslapd-errorlog-logrotationtimeunit***
- ***nsslapd-auditlog-logrotationtime*** and ***nsslapd-auditlog-logrotationtimeunit***
- ***nsslapd-auditfaillog-logrotationtime*** and ***nsslapd-auditfaillog-logrotationtimeunit***

Additionally, you can set the time when the log file is rotated using the following parameters:

- ***nsslapd-accesslog-logrotationsynchour*** and ***nsslapd-accesslog-logrotationsyncmin***
- ***nsslapd-errorlog-logrotationsynchour*** and ***nsslapd-errorlog-logrotationsyncmin***
- ***nsslapd-auditlog-logrotationsynchour*** and ***nsslapd-auditlog-logrotationsyncmin***
- ***nsslapd-auditfaillog-logrotationsynchour*** and ***nsslapd-auditfaillog-logrotationsyncmin***

For details, see the parameter descriptions in the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

Each log file starts with a title, which identifies the server version, host name, and port, for ease of archiving or exchanging log files. For example:

```
389-Directory/1.4.0.11 B2018.197.1151
server.example.com:389 (/etc/dirsrv/slapd-instance)
```

#### 21.3.4.1. Defining a Log File Rotation Policy Using the Command Line

Use the **dsconf config replace** command to modify parameters controlling the Directory Server logging features. For example for the error log, to set access mode **600**, to keep maximum **2**, and to rotate log files at a size of **100** MB or every **5 days**, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-errorlog-
mode=600 nsslapd-errorlog-maxlogsperdir=2 nsslapd-errorlog-maxlogsize=100 nsslapd-errorlog-
logrotationtime=5 nsslapd-errorlog-logrotationtimeunit=day
```

#### 21.3.4.2. Defining a Log File Rotation Policy Using the Web Console

See [Section 21.3.1.2, “Enabling or Disabling Logging Using the Web Console”](#).

#### 21.3.5. Defining a Log File Deletion Policy

Directory Server automatically deletes old archived log files, if you set a **Deletion Policy**.



### NOTE

You can only set a log file deletion policy if you have a log file rotation policy set. Directory Server applies the deletion policy at the time of log rotation.

You can set the following configuration parameters to control the log file deletion policy:

#### Total log size

If the size of all access, error, audit or audit fail log files increases the configured value, the oldest log file is automatically deleted.

- Access log: ***nsslapd-accesslog-logmaxdiskspace***
- Error log: ***nsslapd-errorlog-logmaxdiskspace***
- Audit log: ***nsslapd-auditlog-logmaxdiskspace***
- Audit log: ***nsslapd-auditfaillog-logmaxdiskspace***

#### Free disk space is less than

When the free disk space reaches this value, the oldest archived log file is automatically deleted.

- Access log: ***nsslapd-accesslog-logminfreediskspace***
- Error log: ***nsslapd-errorlog-logminfreediskspace***
- Audit log: ***nsslapd-auditlog-logminfreediskspace***
- Audit log: ***nsslapd-auditfaillog-logminfreediskspace***

#### When a file is older than a specified time

When a log file is older than the configured time, it is automatically deleted.

- Access log: ***nsslapd-accesslog-logexpirationtime*** and ***nsslapd-accesslog-logexpirationtimeunit***
- Error log: ***nsslapd-errorlog-logminfreediskspace*** and ***nsslapd-errorlog-logexpirationtimeunit***
- Audit log: ***nsslapd-auditlog-logminfreediskspace*** and ***nsslapd-auditlog-logexpirationtimeunit***
- Audit log: ***nsslapd-auditfaillog-logminfreediskspace*** and ***nsslapd-auditfaillog-logexpirationtimeunit***

For further details, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).

#### 21.3.5.1. Configuring a Log Deletion Policy Using the Command Line

Use the **dsconf config replace** command to modify parameters controlling the Directory Server logging features. For example, to auto-delete the oldest access log file if the total size of all access log files increases **500 MB**, run:

```
dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-accesslog-logmaxdiskspace=500
```

### 21.3.5.2. Configuring a Log Deletion Policy Using the Web Console

See [Section 21.3.1.2, “Enabling or Disabling Logging Using the Web Console”](#).

### 21.3.6. Manual Log File Rotation

The Directory Server supports automatic log file rotation for all three logs. However, it is possible to rotate log files manually if there are no automatic log file creation or deletion policies configured. By default, access, error, audit and audit fail log files can be found in the following location:

```
/var/log/dirsrv/slapd-instance
```

To rotate log files manually:

1. Stop the instance.

```
# dsctl instance_name stop
```

2. Move or rename the log file being rotated so that the old log file is available for future reference.

3. Start the instance:

```
# dsctl instance_name restart
```

### 21.3.7. Configuring the Log Levels

Both the access and the error log can record different amounts of information, depending on the log level that is set.

You can set the following configuration parameters to control the log levels for the:

- Access log: **nsslapd-accesslog-level**
- Error log: **nsslapd-errorlog-level**

For further details and a list of the supported log levels, see the corresponding section in the [Red Hat Directory Server Configuration, Command, and File Reference](#).



#### NOTE

Changing the log level from the default can cause the log file to grow very rapidly. Red Hat recommends not to change the default values without being asked to do so by the Red Hat technical support.

### 21.3.7.1. Configuring the Log Levels Using the Command Line

Use the **dsconf config replace** command to set the log level.

For example, to enable search filter logging (32) and config file processing (64), set the **nsslapd-errorlog-level** parameter to 96 (32 + 64):

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-errorlog-level=96
```

For example, to enable internal access operations logging (4) and logging of connections, operations, and results (256), set the **nsslapd-accesslog-level** parameter to 260 (4 + 256):

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-accesslog-level=260
```

### 21.3.7.2. Configuring the Log Levels Using the Web Console

To configure the access and error log level using the web console:

1. Open the Directory Server user interface in the web console. See [Section 1.4, “Logging Into Directory Server Using the Web Console”](#).
2. Select the instance.
3. To configure:
  - o The access log level:
    1. Open the **Server Settings** → **Logging** → **Access Log** menu.
    2. Select the log levels in the **Access Logging Levels** section. For example:

<b>Access Logging Levels</b>	
	<b>Logging Level</b>
<input checked="" type="checkbox"/>	<b>Default Logging</b>
<input type="checkbox"/>	<b>Internal Operations</b>
<input type="checkbox"/>	<b>Entry Access and Referrals</b>

- o The error log level:
  1. Open the **Server Settings** → **Logging** → **Error Log** menu.
  2. Select the log levels in the **Error Logging Levels** section. For example:

## Error Logging Levels

	Logging Level
<input type="checkbox"/>	Trace Function Calls
<input type="checkbox"/>	Packet Handling
<input type="checkbox"/>	Heavy Trace Output
<input type="checkbox"/>	Connection Management
<input type="checkbox"/>	Packets Sent/Received
<input type="checkbox"/>	Search Filter Processing
<input type="checkbox"/>	Config File Processing
<input type="checkbox"/>	Access Control List Processing
<input type="checkbox"/>	Log Entry Parsing
<input type="checkbox"/>	Housekeeping
<input type="checkbox"/>	Replication
<input type="checkbox"/>	Entry Cache
<input type="checkbox"/>	Plug-ins
<input type="checkbox"/>	Access Control Summary
<input type="checkbox"/>	Nunc-Stans Connection Logging

4. Click **Save**.

### 21.3.7.3. Logging Internal Operations

Several operations cause additional internal operations in Directory Server. For example, if a user deletes an entry, the server runs several internal operations, such as locating the entry and updating groups in which the user was a member. This section explains the format of internal operations log entries. For details about setting the log level, see [Section 21.3.7, “Configuring the Log Levels”](#).

Directory Server provides the following formats of internal operations logging:

#### Server-initiated Internal Operations

Example of an internal operation log entry that was initiated by the server:

```
[14/Jan/2021:09:45:25.814158882 -0400] conn=Internal(0) op=0(0)(0) MOD dn="cn=uniqueid
generator,cn=config"
[14/Jan/2021:09:45:25.822103183 -0400] conn=Internal(0) op=0(0)(0) RESULT err=0 tag=48
nentries=0 etime=0.0007968796
```

For log entries of this type:

- The **conn** field is set to **Internal** followed by **(0)**.

- The ***op*** field is set to **0(0)(nesting\_level)**. For server-initiated internal operations, both the operation ID and internal operation ID are always **0**. For log entries that are not nested, the nesting level is **0**.

## Client-initiated Internal Operations

Example of an internal operation log entry that was initiated by a client:

```
[14/Jan/2021:09:45:14.382918693 -0400] conn=5 (Internal) op= 15(1)(0) SRCH
base="cn=config,cn=userroot,cn=ldbm database,cn=plugins,cn=config" scope=1
filter="objectclass=vlvsearch" attrs=ALL
[14/Jan/2021:09:45:14.383191380 -0400] conn=5 (Internal) op= 15(1)(0) RESULT err=0 tag=48
nentries=0 etime=0.0000295419
[14/Jan/2021:09:45:14.383216269 -0400] conn=5 (Internal) op= 15(2)(0) SRCH
base="cn=config,cn=NetscapeRoot,cn=ldbm database,cn=plugins,cn=config" scope=1
filter="objectclass=vlvsearch" attrs=ALL
[14/Jan/2021:09:45:14.383449419 -0400] conn=5 (Internal) op= 15(2)(0) RESULT err=0
```

For log entries of this type:

- The ***conn*** field is set to the client connection ID, followed by the string **(Internal)**.
- The ***op*** field contains the operation ID, followed by **(internal\_operation\_ID)(nesting\_level)**. The internal operation ID can vary, and log entries that are not nested, the nesting level is **0**.

If the **nsslapd-plugin-logging** parameter is set to **on** and internal operations logging is enabled, Directory Server additionally logs internal operations of plug-ins.

### Example 21.1. Internal Operations Log Entries with Plug-in Logging Enabled

If you delete the **uid=user,dc=example,dc=com** entry, and the **Referential Integrity** plug-in automatically deletes this entry from the **example** group, the server logs:

```
[time_stamp] conn=2 op=37 DEL dn="uid=user,dc=example,dc=com"
[time_stamp] conn=2 (Internal) op=37(1) SRCH base="uid=user,dc=example,dc=com" scope=0
filter="(|(objectclass=*)(objectclass=ldapsubentry))" attrs=ALL
[time_stamp] conn=2 (Internal) op=37(1) RESULT err=0 tag=48 nentries=1 etime=0.0000129148
[time_stamp] conn=2 (Internal) op=37(2) SRCH base="dc=example,dc=com" scope=2 filter=
(member=uid=user,dc=example,dc=com)" attrs="member"
[time_stamp] conn=2 (Internal) op=37(2) RESULT err=0 tag=48 nentries=0 etime=0.0000123162
[time_stamp] conn=2 (Internal) op=37(3) SRCH base="dc=example,dc=com" scope=2 filter=
(uniquemember=uid=user,dc=example,dc=com)" attrs="uniquemember"
[time_stamp] conn=2 (Internal) op=37(3) RESULT err=0 tag=48 nentries=1 etime=0.0000128104
[time_stamp] conn=2 (Internal) op=37(4) MOD dn="cn=example,dc=example,dc=com"
[time_stamp] conn=2 (Internal) op=37(5) SRCH base="cn=example,dc=example,dc=com"
scope=0 filter="(|(objectclass=*)(objectclass=ldapsubentry))" attrs=ALL
[time_stamp] conn=2 (Internal) op=37(5) RESULT err=0 tag=48 nentries=1 etime=0.0000130685
[time_stamp] conn=2 (Internal) op=37(4) RESULT err=0 tag=48 nentries=0 etime=0.0005217545
[time_stamp] conn=2 (Internal) op=37(6) SRCH base="dc=example,dc=com" scope=2 filter=
(owner=uid=user,dc=example,dc=com)" attrs="owner"
[time_stamp] conn=2 (Internal) op=37(6) RESULT err=0 tag=48 nentries=0 etime=0.0000137656
[time_stamp] conn=2 (Internal) op=37(7) SRCH base="dc=example,dc=com" scope=2 filter=
(seeAlso=uid=user,dc=example,dc=com)" attrs="seeAlso"
[time_stamp] conn=2 (Internal) op=37(7) RESULT err=0 tag=48 nentries=0 etime=0.0000066978
[time_stamp] conn=2 (Internal) op=37(8) SRCH base="o=netscaperoot" scope=2 filter=
```

```
(member=uid=user,dc=com)" attrs="member"
[time_stamp] conn=2 (Internal) op=37(8) RESULT err=0 tag=48 nentries=0 etime=0.0000063316
[time_stamp] conn=2 (Internal) op=37(9) SRCH base="o=netscaperoot" scope=2 filter=
(uniquemember=uid=user,dc=com)" attrs="uniquemember"
[time_stamp] conn=2 (Internal) op=37(9) RESULT err=0 tag=48 nentries=0 etime=0.0000048634
[time_stamp] conn=2 (Internal) op=37(10) SRCH base="o=netscaperoot" scope=2 filter=
(owner=uid=user,dc=com)" attrs="owner"
[time_stamp] conn=2 (Internal) op=37(10) RESULT err=0 tag=48 nentries=0 etime=0.0000048854
[time_stamp] conn=2 (Internal) op=37(11) SRCH base="o=netscaperoot" scope=2 filter=
(seeAlso=uid=user,dc=com)" attrs="seeAlso"
[time_stamp] conn=2 (Internal) op=37(11) RESULT err=0 tag=48 nentries=0 etime=0.0000046522
[time_stamp] conn=2 op=37 RESULT err=0 tag=107 nentries=0 etime=0.0010297858
```

## 21.4. GETTING ACCESS LOG STATISTICS

The **logconv.pl** script parses the access log and returns summary information on different users and operations that have been run on the server.

At its simplest, the script simply parses the access log (or logs):

```
# logconv.pl /relative/path/to/accessLog
```

The script can accept wildcards to parse multiple access logs, which is useful if log rotation is used.

```
# logconv.pl /var/log/dirsrv/slappd-*access*
```

The different options for **logconv.pl** are covered in the manpage and in the *Configuration, Command, and File Reference*.

There are several different ways that **logconv.pl** can be used to pull general usage information from the access logs.

At its simplest, **logconv.pl** prints a list of total operations, total number of connections, counts per each operation type, counts for some extended operations like persistent searches, and bind information.

```
# logconv.pl /var/log/dirsrv/slappd-*access
Access Log Analyzer 8.2
Command: logconv.pl /var/log/dirsrv/slappd-*access
Processing 1 Access Log(s)...

[001] /var/log/dirsrv/slappd-*access size (bytes): 77532
```

Total Log Lines Analysed: 527

Start of Logs: 14/Oct/2017:16:15:22.452909568  
End of Logs: 14/Oct/2017:16:39:50.157790196

Processed Log Time: 0 Hours, 24 Minutes, 27.704877056 Seconds

Restarts: 10  
Secure Protocol Versions:  
- TLS1.2 client bound as uid=*user\_name*,ou=people,o=*example.com* (11 connections)  
- TLS1.2 128-bit AES; client CN=CA Subsystem,O=*example.com*; issuer CN=Certificate

Authority,O=example.com (11 connections)  
 - TLS1.2 128-bit AES-GCM (2 connections)  
 - TLS1.2 128-bit AES (3 connections)

Peak Concurrent Connections: 38

Total Operations: 4771

Total Results: 4653

Overall Performance: 97.5%

Total Connections: 249 (0.17/sec) (10.18/min)  
 - LDAP Connections: 107 (0.07/sec) (4.37/min)  
 - LDAPI Connections: 128 (0.09/sec) (5.23/min)  
 - LDAPS Connections: 14 (0.01/sec) (0.57/min)  
 - StartTLS Extended Ops: 2 (0.00/sec) (0.08/min)

Searches: 2963 (2.02/sec) (121.13/min)  
 Modifications: 649 (0.44/sec) (26.53/min)  
 Adds: 785 (0.53/sec) (32.09/min)  
 Deletes: 10 (0.01/sec) (0.41/min)  
 Mod RDNs: 6 (0.00/sec) (0.25/min)  
 Compares: 0 (0.00/sec) (0.00/min)  
 Binds: 324 (0.22/sec) (13.25/min)

Proxied Auth Operations: 0

Persistent Searches: 17

Internal Operations: 0

Entry Operations: 0

Extended Operations: 4

Abandoned Requests: 0

Smart Referrals Received: 0

VLV Operations: 30

VLV Unindexed Searches: 0

VLV Unindexed Components: 20

SORT Operations: 22

Entire Search Base Queries: 12

Paged Searches: 2

Unindexed Searches: 0

Unindexed Components: 149

FDs Taken: 249

FDs Returned: 212

Highest FD Taken: 107

Broken Pipes: 0

Connections Reset By Peer: 0

Resource Unavailable: 0

Max BER Size Exceeded: 0

Binds: 324

Unbinds: 155

- LDAP v2 Binds: 41
- LDAP v3 Binds: 180
- AUTOBINDs(LDAP): 103

```
- SSL Client Binds:      0
- Failed SSL Client Binds:  0
- SASL Binds:          134
- EXTERNAL:           114
- GSSAPI:              20
- Directory Manager Binds: 10
- Anonymous Binds:       1
```

Cleaning up temp files...

Done.

In addition to the summary information for operations and connections, more detailed summary information for all of the connections to the server. This information includes things like most common IP addresses used to connect to the server, DNs with the most failed login attempts, total bind DNs used to access the server, and the most common error or return codes.

Additional connection summaries are passed as a single option. For example, listing the number of DNs used to connect to the server (**b**) and the total connection codes returned by the server (**c**) are passed as **-bc**.

```
# logconv.pl -bc /var/log/dirsrv/slapd-instance/access
...
----- Total Connection Codes -----

U1      3  Cleanly Closed Connections
B1      1  Bad Ber Tag Encountered

----- Top 20 Bind DN's -----

Number of Unique Bind DN's: 212

1801    cn=Directory Manager
1297    Anonymous Binds
311     uid=jsmith,ou=people...
87      uid=bjensen,ou=peopl...
85      uid=mreynolds,ou=peo...
69      uid=jrockford,ou=peo...
55      uid=sspencer,ou=peop...
...
```

The data can be limited to entries after a certain start time (**-S**), before a certain end time (**-E**), or within a range. When start and end times are set, the **logconv.pl** first prints the time range given, then the summary for that period.

```
# logconv.pl -S "[01/Jul/2016:16:11:47.000000000 -0400]" -E "[01/Jul/2016:17:23:08.999999999 -0400]" /var/log/dirsrv/slapd-instance/access
...
----- Access Log Output -----


Start of Logs: 01/Jul/2016:16:11:47
End of Logs:   01/Jul/2016:17:23:08
...
```

The start and end period only sets time limits for the data used to generate the total summary counts. It still shows aggregated, or total, counts. To get a view of the patterns in connections and operations to

the Directory Server, it is possible to output data with counts per minute (**-M**) or per second (**-m**). In this case, the data are printed, in time unit increments, to a specified CSV output file.

```
# logconv.pl -m|-M outputFile accessLogFile
```

For example:

```
# logconv.pl -M /home/output/statsPerMin.txt /var/log/dirsrv/slapd-instance/access*
```

The **-M|-m** options can also be used with the **-S** and **-E** arguments, to get per-minute or per-second counts within a specific time period.

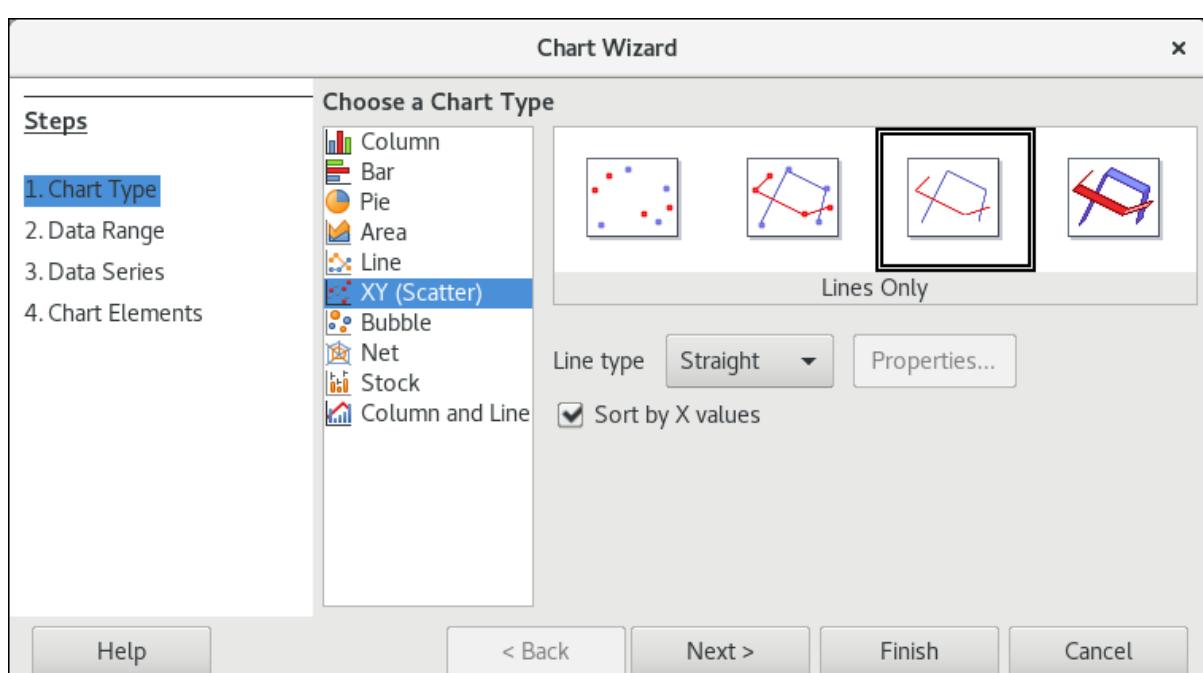
Each row in the file represents one unit of time, either minute or second, with total counts for that time period. The CSV file (for both per-minute and per-second statistics) contains the following columns, in order:

```
Time,time_t,Results,Search,Add,Mod,Modrdn,Delete,Abandon,Connections,SSL Conns,Bind,Anon Bind,Unbind,Unindexed
```

The CSV file can be manipulated in any spreadsheet program, like LibreOffice Calc, and in many other business applications. The procedures for importing the CSV data and generating charts or other metrics depends on the application itself.

For example, to create a chart in LibreOffice Calc:

1. Open the CSV file.
2. Click the **Insert** menu, and select **Chart**.
3. In the **Chart Type** area, set the chart type to **XY (Scatter)**.
  1. Set the subtype to lines only.
  2. Select the option to sort by X values.



4. Accept the defaults in the other screens (particularly, to use the data series in columns and to set the first row and first column as labels), and create the chart.

## 21.5. MONITORING THE LOCAL DISK FOR GRACEFUL SHUTDOWN

See the [Monitoring the Local Disk for Graceful Shutdown](#) section in the *Red Hat Directory Server Performance Tuning Guide*.

## 21.6. MONITORING SERVER ACTIVITY

See the [Monitoring Server Activity](#) section in the *Red Hat Directory Server Performance Tuning Guide*.

## 21.7. MONITORING DATABASE ACTIVITY

See the [Monitoring Database Activity](#) section in the *Red Hat Directory Server Performance Tuning Guide*.

## 21.8. MONITORING DATABASE LINK ACTIVITY

See the [Monitoring Database Link Activity](#) section in the *Red Hat Directory Server Performance Tuning Guide*.

## 21.9. ENABLING AND DISABLING COUNTERS

The **nsslapd-counters** parameter enabled counters to run. However, running counters can affect performance, so it is also possible to turn off counters. If counters are off, they all have a value of zero (0).

By default, counters are already enabled. To enable or disable performance counters, use **ldapmodify**. For example, to disable:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-counters=off
```

## 21.10. MONITORING DIRECTORY SERVER USING SNMP

The server and database activity monitoring log setup described in [Chapter 21, Monitoring Server and Database Activity](#) is specific to Directory Server. You can also monitor your Directory Server using 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.

Directory Server can be monitored with SNMP through an AgentX subagent. SNMP monitoring collects useful information about the Directory Server, such as bind information, operations performed on the server, and cache information. The Directory Server SNMP subagent supports SNMP traps to send notifications about changes in the running state of your server instances.

### 21.10.1. About SNMP

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 is 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 usually runs on the same host machine as the subagents it talks to, although it can run on a remote machine.

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 units (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.

### 21.10.2. Enabling and Disabling SNMP Support

By default, the SNMP protocol is enabled in Directory Server and, after configuring the subagent, you can use it.

To enable or disable SNMP in an instance, set the ***nsSNMPEnabled*** parameter to **on** or **off**. For example, to disable SNMP in a Directory Server instance:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=SNMP,cn=config
changetype: modify
replace: nsSNMPEnabled
nsSNMPEnabled: on
```

### 21.10.3. Setting Parameters to Identify an Instance Using SNMP

Directory Server provides the following attributes which help identifying instances using SNMP:

- ***nsSNMPOrganization***
- ***nsSNMPLocation***
- ***nsSNMPContact***
- ***nsSNMPDescription***

For details about the parameters, see their descriptions in the [cn=SNMP](#) section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to set the ***nsSNMPLocation*** parameter to **Munich, Germany**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=SNMP,cn=config
changetype: modify
replace: nsSNMPLocation
nsSNMPLocation: Munich, Germany
```

## 21.10.4. Setting up an SNMP Agent for Directory Server

To query information from Directory Server using the SNMP protocol, set up an SNMP agent:

1. Install the 389-ds-base-snmp and net-snmp packages:

```
# yum install 389-ds-base-snmp net-snmp
```

2. To configure the SNMP master agent, edit the **/etc/snmp/snmpd.conf** file, adding the following entry to enable the agent extensibility (AgentX) protocol:

```
master agentx
```

For further details about the AgentX protocol, see [RFC 2741](#).

3. To configure the SNMP subagent, edit the **/etc/dirsrv/config/ldap-agent.conf** file, adding a server parameter for each Directory Server instance you want to monitor. For example:

```
server slapd-instance_name
```

4. Optionally, create an SNMP user account:

- a. Stop the **snmpd** service:

```
# systemctl stop snmpd
```

- b. Create the SNMP user account. For example:

```
# net-snmp-create-v3-user -A authentication_password -a SHA \
-X private_password -x AES user_name
```

For details about the parameters used in the command, see the `net-snmp-create-v3-user(1)` man page.

- c. Start the **snmpd** service:

```
# systemctl start snmpd
```

5. Optionally, set the Directory Server descriptive properties. For details, see [Section 21.10.3, “Setting Parameters to Identify an Instance Using SNMP”](#).

6. Start the **dirsrv-snmp** service:

```
# systemctl start dirsrv-snmp
```

7. Optionally, to verify the configuration:

- a. Install the net-snmp-utils package:

```
# yum install net-snmp-utils
```

- b. Query the Directory Server Object Identifiers (OID). For example:

```
# snmpwalk -v3 -u user_name -M /usr/share/snmp/mibs:/usr/share/dirsrv/mibs/ \
-I AuthPriv -m +RHDS-MIB -A authentication_password -a SHA
-X private_password -x AES server.example.com .1.3.6.1.4.1.2312.6.1.1
```

## 21.10.5. Configuring SNMP Traps

An SNMP trap is essentially a threshold which triggers a notification if it is encountered by the monitored server. To use traps, the master agent must be configured to accept traps and do something with them. For example, a trap can trigger an email notification for an administrator of the Directory Server instance stops.

The subagent is only responsible for sending the traps to the master agent. The master agent and a trap handler must be configured according to the documentation for the SNMP master agent you are using.

Traps are accompanied by information from the **Entity Table**, which contains information specific to the Directory Server instance, such as its name and version number. The **Entity Table** is described in [Section 21.10.6.3, “Entity Table”](#). This means that the action the master agent takes when it receives a trap is flexible, such as sending an email to an email address defined in the **dsEntityContact** variable for one instance while sending a notification to a pager number in the **dsEntityContact** variable for another instance.

There are two traps supported by the subagent:

- *DirectoryServerDown*. This trap is generated whenever the subagent detects the Directory Server is potentially not running. This trap will be sent with the Directory Server instance description, version, physical location, and contact information, which are detailed in the **dsEntityDescr**, **dsEntityVers**, **dsEntityLocation**, and **dsEntityContact** variables.
- *DirectoryServerStart*. This trap is generated whenever the subagent detects that the Directory Server has started or restarted. This trap will be sent with the Directory Server instance description, version, physical location, and contact information, which are detailed in the **dsEntityDescr**, **dsEntityVers**, **dsEntityLocation**, and **dsEntityContact** variables.

## 21.10.6. Using the Management Information Base

The Directory Server's MIB is a file called **redhat-directory.mib** stored in the **/usr/share/dirsrv/mibs** directory. 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 [Section 21.10.4, “Setting up an SNMP Agent for Directory Server”](#).

The client tools need to load the Directory Server MIB to use the variable names listed in the following sections.

Using the directory MIB enables administrators to use SNMP to see administrative information about the directory and monitor the server in real-time. The directory MIB is broken into four distinct tables of managed objects:

- [Section 21.10.6.1, "Operations Table"](#)
- [Section 21.10.6.2, "Entries Table"](#)
- [Section 21.10.6.3, "Entity Table"](#)
- [Section 21.10.6.4, "Interaction Table"](#)



### NOTE

All of the Directory Server attributes monitored by SNMP use 64-bit integers for the counters, even on 32-bit systems.

#### 21.10.6.1. Operations Table

The **Operations Table** provides statistical information about Directory Server access, operations, and errors. [Table 21.1, "Operations Table: Managed Objects and Descriptions"](#) describes the managed objects stored in the **Operations Table** of the **redhat-directory.mib** file.

**Table 21.1. Operations Table: 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 TLS 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 <b>0</b> because LDAP implements read operations indirectly using the search operation.
dsCompareOps	The number of compare operations serviced by this directory since server startup.

Managed Object	Description
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 <b>0</b> because LDAP implements list operations indirectly using 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.

### 21.10.6.2. Entries Table

The **Entries Table** provides information about the contents of the directory entries. [Table 21.2, “Entries Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the **Entries Table** in the **redhat-directory.mib** file.

**Table 21.2. Entries Table: Managed Objects and Descriptions**

Managed Object	Description

Managed Object	Description
dsCopyEntries	The number of directory entries for which this directory contains a copy. The value of this object will always be <b>0</b> (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.

### 21.10.6.3. Entity Table

The **Entity Table** contains identifying information about the Directory Server instance. The values for the **Entity Table** are set in **cn=SNMP,cn=config** entry as described in [Section 21.10.3, “Setting Parameters to Identify an Instance Using SNMP”](#).

[Table 21.3, “Entity Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the **Entity Table** of the **redhat-directory.mib** file.

**Table 21.3. Entity Table: Managed Objects and Descriptions**

Managed Object	Description
dsEntityDescr	The description set for the Directory Server instance.
dsEntityVers	The Directory Server version number of the Directory Server instance.
dsEntityOrg	The organization responsible for the Directory Server instance.
dsEntityLocation	The physical location of the Directory Server instance.
dsEntityContact	The name and contact information for the person responsible for the Directory Server instance.
dsEntityName	The name of the Directory Server instance.

### 21.10.6.4. Interaction Table



#### NOTE

The **Interaction Table** is *not* supported by the subagent. The subagent can query the table, but it will not ever be updated with valid data.

[Table 21.4, “Interaction Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the **Interaction Table** of the **redhat-directory.mib** file.

**Table 21.4. Interaction Table: Managed Objects and Descriptions**

Managed Object	Description
dsIntTable	Details, in each row of the table, 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.
dsIntIndex	Part of the unique key, together with <b>applIndex</b> , to identify the conceptual row which contains useful information on the (attempted) interaction between the Directory Server (referred to by <b>applIndex</b> ) and a peer Directory Server.
dsName	The distinguished name (DN) of the peer Directory Server to which this entry belongs.
dsTimeOfCreation	The value of <b>sysUpTime</b> 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 <b>sysUpTime</b> 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 <b>sysUpTime</b> 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 22. MAKING A HIGH-AVAILABILITY AND DISASTER RECOVERY PLAN

Part of running a Directory Server deployment efficiently is planning for that worst case scenario. This chapter covers general principles for drafting a disaster recovery plan and highlights features in Directory Server that can be used to aide in disaster recovery.

*Disaster recovery* is a way of planning and implementing a smooth transition from one operating environment to another environment whenever there is some sort of catastrophic failure. A disaster recovery plan for Directory Server may be part of a larger business continuity plan or it could be a standalone plan specifically for an interruption in directory services.



## NOTE

This chapter covers very general concepts for disaster recovery.

Disaster recovery can be a very complex and detail-specific thing. Consider using a professional service to design, maintain, and test any disaster recovery plan for sensitive or mission-critical services, like Red Hat Directory Server.

## 22.1. IDENTIFYING POTENTIAL SCENARIOS

The first step is identifying what potential issues you may encounter, what services will be affected, and what responses you should take. In the *Red Hat Directory Server Deployment Guide*, administrators made a site survey of their existing and proposed infrastructure to determine what kind of directory to design. Do something similar for disaster planning; as in [Table 22.1, “Disaster Scenarios and Responses”](#), identify where your data infrastructure is, determine what the affect of losing that component is, and look at potential ideal responses.

**Table 22.1. Disaster Scenarios and Responses**

Scenario	Effects on Infrastructure	Ideal Response
Data corruption	Through software or hardware failure (or through a malicious attack), the data at one site or on one server could be corrupted. If that corrupted server is a supplier in multi-supplier replication, then the corruption can quickly be propagated throughout the deployment.	An isolated server should be available with access to the most recent backup of uncorrupted data. When a problem is detected, replication can be suspended on the regular infrastructure, and this server can be brought online to reinitialize the suppliers with good data.
Natural disasters and other mass events	Natural disasters can take an entire office or data center offline, even through something as simple as a long-term power outage.	Directory operations can be transferred to a mirrored site at another physical location, with the same data.
Server or machine loss	A single machine could fail.	Another machine, with the same data, can assume the lost machine's place.

## 22.2. DEFINING THE TYPE OF ROLLOVER

Disaster recovery, as the introduction says, is the process for transitioning from one system to another system with as little interruption of service as possible. That's called a *rollover*, and there are three different ways of doing a rollover:

- A *hot* rollover means that the infrastructure is completely mirrored at another site and that the backup site is always up and current with the primary site. This requires only a few adjustments to switch operations from the primary to the backup.
- A *warm* rollover means that all of the elements for the backup site are in place (adequate network connections, all required applications and hardware) but the system is not actively running or necessarily configured. This can require some extra time to configure the machines and get the system running.
- A *cold* rollover means that a site is available but there are few resources immediately available to set it up.

The obvious difference in the types of rollover is the time and expense necessary to set up the backup site. Hot and warm sites have higher initial expenditures to set up and run.

A mix of rollover types can be used, depending on the specific disaster scenario being planned. For example, a rollover plan for the loss of a single server could use a hot rollover easily and relatively cheaply by creating and keeping a virtual machine copy of the Directory Server instance which can be brought online within minutes. It would not even require keeping the virtual machine in a separate facility or network. On the other hand, a cold rollover could be planned for the loss of an entire data center or office.

Match the rollover process to the severity of the disaster scenario, your budget and available resources, and the likelihood of encountering problems.

## 22.3. IDENTIFYING USEFUL DIRECTORY SERVER FEATURES FOR DISASTER RECOVERY

The hardest part of a recovery is not the hardware; it is getting a reliable copy of the data in the server. There are three Directory Server features that are excellent tools for preparing data copies for disaster recovery:

- Backing up databases and verifying the backups regularly
- Multi-supplier replication, chaining, backing up databases, and monitoring the server with a named pipe script
- Chaining

Additionally, monitoring the server with a named pipe script and with other Directory Server performance counters can be effective at catching and quickly responding to specific, critical events.

### 22.3.1. Backing up Directory Data for Disaster Recovery

The most useful tool for disaster recovery is to do frequent backups of a directory instance. Archives can be stored on physical media, at different locations than the primary data center or on-site at a cold backup location.

Backups can be automated to run regularly through cron jobs. For example, to create a backup of the **ldap://server.example.com** instance daily at 22:00 (10pm):

```
0 22 * * 1 /usr/sbin/dsconf -D "cn=Directory Manager" ldap://server.example.com backup create
```

The **dsconf backup create** command backs up the directory data without having to stop the server first.



#### NOTE

Red Hat recommends to back up the data on all servers in a multi-supplier replication environment.

Backing up both directory databases and the directory configuration (**dse.ldif** file) are covered in [Section 6.3, “Backing up Directory Server”](#).

### 22.3.2. Multi-Supplier Replication for High-availability

Multi-supplier replication is the best defense against losing a single server and, possibly, even an entire office or department. While a small number of servers are data suppliers, multiple servers all hold the same data – potentially dozens of suppliers and hubs in a single replication environment. This keeps information accessible to clients even if multiple servers go offline.

Replication can be used to copy over data to servers and bring replacements online more quickly.



#### NOTE

To protect against data corruption being propagated through replication, frequently back up the database.

Replication configuration also allows write operations to be referred to failover servers if the primary supplier is inaccessible. This means that write operations can proceed as normal from the client perspective, even when servers go offline.

#### Example 22.1. Scenarios for Multi-Supplier Replication

Replication is a versatile tool for disaster recovery in several scenarios:

- For a single server failure, all of the data stored on that instance is both accessible and retrievable from other servers.
- For the loss of an entire office or colocation facility, servers can be mirrored at an entirely different physical location (which is aided by Directory Server's wide area replication performance). With minimal effort, traffic can be redirected to the replicated site without having to bring new servers online.

Configuring replication is covered in [Chapter 15, Managing Replication](#).

### 22.3.3. Chaining Databases for High-availability

*Chaining* is a configuration where a client sends a request to one server and it automatically forwards that request to another server to process. There can be multiple servers configured in the database link (or chain) to allow for automatic failover if one server is not available.

### Example 22.2. Scenarios for Chaining

When chaining is combined with a list of failover servers, client traffic can be automatically redirected from a single server (or even group of servers) when they are offline. This does not help in recovery, but it helps manage the transition from primary to backup servers.

Chaining databases is covered in [Section 2.3, "Creating and Maintaining Database Links"](#).

## 22.4. DEFINING THE RECOVERY PROCESS

There are a lot of tools that can help with disaster recovery, but an effective recovery process circles back to having a well-defined plan of what to do in every scenario. Two things, at least, need to be clearly identified:

- What signals a disaster? Some things are obvious (a massive power outage, network loss, or fire), but other situations need to be defined. For example, what signals that a backup server needs to be brought online?
- Who responds to a disaster and how? Once a disaster situation occurs, who has the responsibility to act? How are they notified of the event? What are they expected to do?



### IMPORTANT

- Store a printed copy off the disaster recovery plan off-site.
- Test the disaster recovery plan on a regular basis and after configuration and infrastructure changes.

## 22.5. BASIC EXAMPLE: PERFORMING A RECOVERY

An administrator, John Smith, has to create a disaster recovery plan for his directory deployment. Example Corp. has three physical offices, in San Francisco, Dallas, and Arlington. Each site has 10 servers which replicate to each other locally, and then one server at each site replicates to another server at the other two sites.

Each site has business-critical customer data stored in its directory, as well as human resources data. Several external applications require access to the data to perform operations like billing.

John Smith's first step is to perform a site survey. He is looking for three things: what his directory usage is (clients that access it and traffic loads across the sites), what his current assets are, and what assets he may need to acquire. This is much like the initial site survey he performed when deploying Red Hat Directory Server.

His next step is identifying potential disaster scenarios. Two of the three sites are highly vulnerable to natural disasters (San Francisco and Dallas). All three sites could face normal interruptions, like outages for power or Internet access. Additionally, since each site supplies its own local data, each site is vulnerable to losing a server instance or machine.

John Smith then breaks his disaster recovery plan into three parts:

- Plan A covers losing a single instance of Directory Server
- Plan B covers some kind of data corruption or attack
- Plan C covers losing an entire office

For plans A and B, John Smith decides to use a hot recovery to immediately switch functionality from a single instance to the backup. Each server is backed up daily, using a cron job, and then the archive is copied over and restored on a virtual machine. The virtual machine is kept on a different subnet, but can be switched over immediately if its peer ever does offline. John Smith uses simple SNMP traps to track each Directory Server instance's availability.

Plan C is more extensive. Along with replication between sites and the local backups, he decides to mail a physical copy of each site's backup, for every local instance, once a week to the other two colocation facilities. He also keep a spare server with adequate Internet access and software licenses to restore an entire site, using virtual machines, one of the other different colocation facilities. He designates the Arlington site as the primary recovery location because that is where most of the IT staff is located, then San Francisco and last Dallas, based on the distribution of personnel. For every event, the IT administrator at all three sites will be notified, and the manager assumes the responsibilities of setting up the virtual machines, restoring the Directory Server instances from the physical backups, and rerouting client traffic.

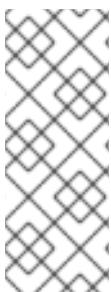
John Smith schedules to review and update the plan quarterly to account for any new hardware or application changes. Once a year, all three sites have to run through the procedure of recovering and deploying the other two sites, according to the procedures in Disaster Plan C.

# CHAPTER 23. CREATING TEST ENTRIES

The **dsctl Idifgen** command creates LDIF files with different types of test entries. For example, you can use this LDIF file to populate a test instance or a sub-tree to test the performance of Directory Server with the example entries.

You can pass one of the following entry type arguments to **dsctl Idifgen**:

- **users**: Creates an LDIF file that contains user entries.
- **groups**: Creates an LDIF file that contains static group and member entries.
- **cos-def**: Creates an LDIF file that either contains a classic pointer or an indirect Class of Service (CoS) definition.
- **cos-template**: Creates an LDIF file that contains a CoS template.
- **roles**: Creates an LDIF file that contains managed, filtered, or indirect role entries.
- **mod-load**: Creates an LDIF file that contains modify operations. Import this file using the **ldapmodify** utility.
- **nested**: Creates an LDIF file that contains heavily nested entries in a cascading or fractal tree design.



## NOTE

The **dsctl Idifgen** command creates only the LDIF file. To load the file into your Directory Server instance, use the:

- **ldapmodify** utility after you created an LDIF file using the **mod-load** option
- **ldapadd** utility for all other options

Except for the **nested** entry type, if you do not provide any command line options, the **dsctl Idifgen** command uses an interactive mode:

```
# dsctl instance_name Idifgen entry_type
```

## 23.1. CREATING AN LDIF FILE WITH EXAMPLE USER ENTRIES

Use the **dsctl Idifgen users** command to create an LDIF file with example user entries. For example, to create an LDIF file named **/tmp/users.Idif** that adds 100,000 generic users to the **dc=example,dc=com** suffix, enter:

```
# dsctl instance_name Idifgen users --suffix "dc=example,dc=com" --number 100000 --generic --ldif-file=/tmp/users.Idif
```

Note that the command creates the following organizational units (OU) and randomly assigns the users to these OUs:

- **ou=accounting**
- **ou=product development**

- **ou=product testing**
- **ou=human resources**
- **ou=payroll**
- **ou=people**
- **ou=groups**

For further details and other options you can use to create the LDIF file, enter:

```
# dsctl instance_name ldifgen users --help
```

## 23.2. CREATING AN LDIF FILE WITH EXAMPLE GROUP ENTRIES

Use the **dsctl ldifgen groups** command to create an LDIF file with example group entries. For example, to create an LDIF file named **/tmp/groups.ldif** that adds 500 groups to the **ou=groups,dc=example,dc=com** entry, and each group has has **100** members, enter:

```
# dsctl instance_name ldifgen groups --number 500 --suffix "dc=example,dc=com" --parent "ou=groups,dc=example,dc=com" --num-members 100 --create-members --member-parent "ou=People,dc=example,dc=com" --ldif-file /tmp/group.ldif example
```

Note that the command also creates LDIF statements to add the user entries in **ou=People,dc=example,dc=com**.



### IMPORTANT

If you create large groups and try to add the group using the **ldapmodif** utility, you can exceed the maximum Basic Encoding Rules (BER) size limit, and the import fails. In this case, increase the value of the **nsslapd-maxbersize** parameter in the **cn=config** entry.

For further details and other options you can use after you create the LDIF file, enter:

```
# dsctl instance_name ldifgen groups --help
```

## 23.3. CREATING AN LDIF FILE WITH AN EXAMPLE COS DEFINITION

Use the **dsctl ldifgen cos-def** command to create an LDIF file with a Class of Service (CoS) definition. For example, to create an LDIF file named **/tmp/cos-definition.ldif** that adds a classic CoS definition to the **ou=cos definitions,dc=example,dc=com** entry:

```
# dsctl instance_name ldifgen cos-def Postal_Def --type classic --parent "ou=cos definitions,dc=example,dc=com" --cos-specifier businessCategory --cos-template "cn=sales,cn=classicCoS,dc=example,dc=com" --cos-attr postalcode telephonenumber --ldif-file /tmp/cos-definition.ldif
```

For further details about the options used in the example and other options you can set to create the LDIF file, enter:

```
# dsctl instance_name ldifgen cos-def --help
```

## 23.4. CREATING AN LDIF FILE WITH EXAMPLE MODIFICATION STATEMENTS

Use the **dsctl ldifgen mod-load** command to create an LDIF file that contains update operations.

```
# dsctl instance_name ldifgen mod-load --parent dc=example,dc=com --num-users 1000 --create-users --mod-users 1000 --add-users 10 --del-users 100 --mod-users 1000 --modrdn-users 100 --mod-attrs cn uid sn --delete-users
```

This command creates the **/tmp/modifications.ldif** file with the statements that do the following:

1. Create an LDIF file with 1000 **ADD** operations to create user entries.
2. Modify all entries by changing their ***cn*, *uid*, *sn*** attributes.
3. Add additional 10 user entries.
4. Perform 100 **MODRDN** operations.
5. Delete 100 entries
6. Delete all remaining entries at the end.

For further details and other options you can use to create the LDIF file, enter:

```
# dsctl instance_name ldifgen mod-load --help
```

## 23.5. CREATING AN LDIF FILE WITH NESTED EXAMPLE ENTRIES

Use the **dsctl ldifgen nested** command to create an LDIF file that contains a heavily nested cascading fractal structure. For example, to create an LDIF file named **/tmp/nested.ldif**, that adds 600 users in total in different organization units (OU) under the **dc=example,dc=com** entry, with each OU containing a maximum number of 100 users:

```
# dsctl instance_name ldifgen nested --num-users 600 --node-limit 100 --suffix "dc=example,dc=com"
```

For further details about the options, enter:

```
# dsctl instance_name ldifgen nested --help
```

## APPENDIX A. USING LDAP CLIENT TOOLS

Red Hat Directory Server uses the LDAP tools (such as **ldapsearch** and **ldapmodify**) supplied with OpenLDAP. The OpenLDAP tool options are described in the OpenLDAP man pages at <http://www.openldap.org/software/man.cgi>.

This appendix gives some common usage scenarios and examples for using these LDAP tools.

More extensive examples for using **ldapsearch** are given in [Chapter 14, Finding Directory Entries](#). More examples for using **ldapmodify** and **ldapdelete** are given in [Chapter 3, Managing Directory Entries](#).

### A.1. RUNNING EXTENDED OPERATIONS

Red Hat Directory Server supports a variety of extended operations, especially extended search operations. An extended operation passes an additional operation (such as a get effective rights search or server-side sort) along with the LDAP operation. Likewise, LDAP clients have the potential to support a number of extended operations.

The OpenLDAP LDAP tools support extended operations in two ways. All client tools (**ldapmodify**, **ldapsearch**, and the others) use either the **-e** or **-E** options to send an extended operation. The **-e** argument can be used with any OpenLDAP client tool and sends general instructions about the operation, like how to handle password policies. The **-E** is used only with **ldapsearches** and passes more useful controls like GER searches, sort and page information, and information for other, not-explicitly-supported extended operations.

Additionally, OpenLDAP has another tool, **ldapexop**, which is used exclusively to perform extended search operations, the same as running **ldapsearch -E**.

The format of an extended operation with **ldapsearch** is generally:

**-E extended\_operation\_type=operation\_parameters**

When an extended operation is explicitly handled by the OpenLDAP tools, then the *extended\_operation\_type* can be an alias, like **deref** for a dereference search or **sss** for server-side sorting. A supported extended operation has formatted output. Other extended operations, like GER searches, are passed using their OID rather than an alias, and then the *extended\_operation\_type* is the OID. For those unsupported operations the tool does not recognize the response from the server, so the output is unformatted.

For example, the **pg** extended operation type formats the results in simple pages:

```
# ldapsearch -x -D "cn=Directory Manager" -W -b "ou=Engineers,ou=People,dc=example,dc=com" -E
pg=3 "(objectclass=*)" cn

dn: uid=jsmith,ou=Engineers,ou=People,dc=example,dc=com
cn: John Smith

dn: uid=bjensen,ou=Engineers,ou=People,dc=example,dc=com
cn: Barbara Jensen

dn: uid=hmartin,ou=Engineers,ou=People,dc=example,dc=com
cn: Henry Martin

Results are sorted.
next page size (3): 5
```

The same operation with **Idapexop** can be run using only the OID of the simple paged results operation and the operation's settings (3 results per page):

```
Idapexop 1.2.840.113556.1.4.319=3
```

However, **Idapexop** does not accept the same range of search parameters that **Idapsearch** does, making it less flexible.

## A.2. COMPARING ENTRIES

**Idapcompare** checks entries to see if the specified entry or entries contain an attribute of a specific value. For example, this checks to see if an entry has an **sn** value of Smith:

```
# ldapcompare -D "cn=Directory Manager" -W -p 389 -h server.example.com -x sn:smith
uid=bjensen,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=bjensen,ou=people,dc=example,dc=com"
compare FALSE

ldapcompare -D "cn=Directory Manager" -W -p 389 -h server.example.com -x sn:smith
uid=jsmith,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=jsmith,ou=people,dc=example,dc=com"
compare TRUE
```

The compare attribute can be specified in one of three ways:

- A single *attribute:value* statement passed in the command line directly

```
sn:Smith
```

- A single *attribute::base64value* statement passed in the command line directly, for attributes like **jpegPhoto** or to verify certificates or CRLs

```
jpegPhoto:dkdkPDKCDdko0eiofk==
```

- An *attribute:file* statement that points to a file containing a list of comparison values for the attribute, and the script iterates through the list

```
postalCode:/tmp/codes.txt
```

The compare operation itself has to be run against a specific entry or group of entries. A single entry DN can be passed through the command line, or a list of DNs to be compared can be given using the **-f** option.

### Example A.1. Comparing One Attribute Value to One Entry

Both the attribute-value comparison and the DN are passed with the script.

```
ldapcompare -D "cn=Directory Manager" -W -p 389 -h server.example.com -x sn:smith
uid=jsmith,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=jsmith,ou=people,dc=example,dc=com"
compare TRUE
```

### Example A.2. Comparing a List Attribute Values from a File

First, create a file of possible ***sn*** values.

```
jensen  
johnson  
johannson  
jackson  
jorgenson
```

Then, create a list of entries to compare the values to.

```
uid=jen200,ou=people,dc=example,dc=com  
uid=dsj,ou=people,dc=example,dc=com  
uid=matthewjms,ou=people,dc=example,dc=com  
uid=john1234,ou=people,dc=example,dc=com  
uid=jack.son.1990,ou=people,dc=example,dc=com
```

Then run the script.

```
# ldapcompare -D "cn=Directory Manager" -W -p 389 -h server.example.com -x  
sn:/tmp/surnames.txt -f /tmp/names.txt  
comparing type: "sn" value: "jensen" in entry "uid=jen200,ou=people,dc=example,dc=com"  
compare TRUE
```

## A.3. CHANGING PASSWORDS

The **ldappasswd** command can either set a new user-defined password or generate a new password for an account. Other settings (for bind information, connection information, or other command settings) may be required and are listed in the OpenLDAP manpages.

```
# ldappasswd -x -D bind_dn -W -p server_port -h server_hostname [-A | -a oldPassword] [-S | -s newPassword] [user]
```



### IMPORTANT

Password change operations must be run over a secure connection, such as TLS, STARTTLS, or SASL. For information on how to configure TLS for LDAP clients, see [Section 9.9.4, “Authenticating Using a Certificate”](#).

### Example A.3. Directory Manager Changing a User's Password Over TLS

The Directory Manager changes the password of the user ***uid=tuser1,ou=People,dc=example,dc=com*** to *new\_password* over TLS.

```
# ldappasswd -D "cn=Directory Manager" -W -ZZ -p 389 -h server.example.com -x -s  
new_password "uid=tuser1,ou=People,dc=example,dc=com"
```

### Example A.4. Directory Manager Generating a User's Password

The Directory Manager generates the password of the user **uid=tuser2,ou=People,dc=example,dc=com** over TLS.

```
# ldappasswd -D "cn=Directory Manager" -W -ZZ -p 389 -h server.example.com -x  
"uid=tuser2,ou=People,dc=example,dc=com"
```

#### Example A.5. User Changing His Own Password

A user, **tuser3**, changes the password from **old\_newpassword** to **new\_password** over TLS.

```
# ldappasswd -p 389 -h server.example.com -ZZ -x -D  
"uid=tuser3,ou=People,dc=example,dc=com" -W -a old_password -s new_password
```

#### Example A.6. User Authenticating with DIGEST\_MD5 and Changing His Password

A user, jsmith, authenticates with GSS-API and changes the password to *new\_password*.

```
# ldappasswd -p 389 -h server.example.com -O noplain,minssf=1,maxbufsize=512 -Y GSSAPI -U  
"dn:uid=jsmith,ou=people,dc=example,dc=com" -R EXAMPLE.COM -W -s new_password
```

#### Example A.7. User Already Authenticated by Kerberos Prompts for a New Password

A user, who has already authenticated by Kerberos, prompts for the new password. This is not performed over TLS.

```
# ldappasswd -p 389 -h server.example.com -O noplain,minssf=1,maxbufsize=512 -I
```

## A.4. GENERATING LDAP URLs

LDAP URLs are used in a variety of different configuration areas and operations: referrals and chaining, replication, synchronization, ACIs, and indexing, as a starting list. Constructing accurate LDAP URLs is critical, because incorrect URLs may connect to the wrong server or simply cause operations to fail. Additionally, all OpenLDAP tools allow the **-H** option to pass an LDAP URL instead of other connection information (like the host name, port, subtree, and search base).



### NOTE

LDAP URLs are described in [Appendix C, "LDAP URLs"](#).

The **ldapurl** command manages URL in two ways:

- Deconstruct a given LDAP URL into its constituent element
- Construct a new, valid LDAP URL from given elements

The parameters for working with URLs are listed in [Table A.1, "ldapurl Parameters"](#); the full list of parameters are in the OpenLDAP manpages.

**Table A.1. Idapurl Parameters**

Option	Description
<b>For Deconstructing a URL</b>	
-H "URL"	Passes the LDAP URL to break down into elements.
<b>For Constructing a URL</b>	
-a attributes	Gives a comma-separated attributes that are specifically returned in search results.
-b base	Sets the search base or subtree for the URL.
-f filter	Sets the search filter to use.
-h hostname	Gives the Directory Server's host name.
-p port	Gives the Directory Server's port.
-S ldap ldaps ldapi	Gives the protocol to use to connect, such as <b>ldap</b> , <b>ldaps</b> , or <b>ldapi</b> .
-s scope	Gives the search scope.

**Example A.8. Deconstructing an LDAP URL**

**Idapurl** uses the **-H** option to feed in an existing LDAP URL, and the tool returns the elements of the URL in a neat list:

```
# Idapurl -H "ldap://:389/dc=example,dc=com?cn,sn?sub?(objectclass=inetorgperson)"
scheme: ldap
port: 389
dn: dc=example,dc=com
selector: cn
selector: sn
scope: sub
filter: (objectclass=inetorgperson)
```

**Example A.9. Constructing an LDAP URL**

The most useful application of **Idapurl** is to construct a valid LDAP URL manually. Using **Idapurl** ensures that the URL is valid.

**Idapurl** accepts the normal connection parameters of all LDAP client tools and additional **Idapsearch** arguments for search base, scope, and attributes, but this tool never connects to a Directory Server instance, so it does not require any bind information. It accepts the connection and search settings and feeds them in as elements to the URL.

```
||  ldapurl -a cn,sn -b dc=example,dc=com -s sub -f "(objectclass=inetorgperson)"
```

```
||  ldap://:389/dc=example,dc=com?cn,sn?sub?(objectclass=inetorgperson)
```

## APPENDIX B. 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, LDIF files can be created using virtually any language. All directory data is stored using the UTF-8 encoding of Unicode. Therefore, the LDIF files created must also be UTF-8 encoded.

For information on using LDIF to modify directory entries, see [Chapter 3, Managing Directory Entries](#).

### B.1. ABOUT THE 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
...
```

- Every LDIF entry must have a DN and at least one object class definition.
- Include any attributes required by the object classes defined for the entry.
- All other attributes and object classes are optional.
- Object classes and attributes can be specified in any order.
- The space after the colon is optional.

[Table B.1, “LDIF Fields”](#) describes the LDIF fields shown in the previous definition.

**Table B.1. LDIF Fields**

Field	Definition
<i>[id]</i>	<i>Optional.</i> A positive decimal number representing the entry ID. The database creation tools generate this ID automatically. Never add or edit this value yourself.
dn: <i>distinguished_name</i>	Specifies the distinguished name for the entry.

Field	Definition
objectClass: <i>object_class</i>	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 <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of standard object classes and <a href="#">Chapter 12, Managing the Directory Schema</a> for information on customizing the schema.
<i>attribute_type</i>	Specifies a descriptive attribute to use with the entry. The attribute should be defined either in the schema. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of standard attributes and <a href="#">Chapter 12, Managing the Directory Schema</a> for information on customizing the schema.
[ <i>subtype</i> ]	<i>Optional.</i> 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 a complete list of the supported subtypes tags, see <a href="#">Table D.1, "Supported Language Subtypes"</a> .
<i>attribute_value</i>	Specifies the attribute value to be used with the attribute type.



## NOTE

The LDIF syntax for representing a change to an entry in the directory is different from the syntax described in [Table B.1, "LDIF Fields"](#). For information on using LDIF to modify directory entries, see [Chapter 3, Managing Directory Entries](#).

## B.2. CONTINUING LINES IN LDIF

In LDIF files, a line can be broken and continued (called *folded*) by indenting the continued portion of the line by exactly one space. For example, the following two statements are identical:

```
dn: cn=some_example_user,dc=example,dc=com
dn: cn=some_e
      xample_user,
      dc=example,d
      c=com
```

It is not required to break and continue LDIF lines. However, doing so may improve the readability of the LDIF file. The usual convention is that an LDIF file does not contain more than 78 columns of text.

## B.3. REPRESENTING BINARY DATA

Binary data, such as a JPEG image, is represented in LDIF using one of two methods, standard LDIF notation or base-64 encoding.

### B.3.1. Standard LDIF Notation

Standard LDIF notation uses the lesser than (<) symbol to indicate that the data are binary. For example:

```
jpegphoto: < file:/path/to/photo
```

With this standard notation, it is not necessary to specify the **ldapmodify -b** parameter. However, standard notation requires that the following line be added to the beginning of the LDIF file or the LDIF update statements:

```
version: 1
```

For example:

```
# ldapmodify -x -D userDN -W

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

### B.3.2. Base-64 Encoding

Binary data can be converted to base-64, which can be used in LDIF files, for a variety of data, from images to TLS certificates. Base 64-encoded data are identified by using the :: symbol. For example:

```
jpegPhoto::encoded_data
```

In addition to binary data, other values that must be base-64 encoded include the following:

- 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
```

*attribute\_name* is the name of the attribute to which the binary data is supplied. The binary data is read from standard input and the results are written to standard output. Thus, 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 ***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.

## B.4. SPECIFYING DIRECTORY ENTRIES USING LDIF

Many types of entries can be stored in the directory. This section concentrates on three of the most common types of entries used in a directory: domain, organizational unit, and organizational person entries.

The object classes defined for an entry are what indicate whether the entry represents a domain or domain component, an organizational unit, an organizational person, or some other type of entry. For a complete list of the object classes that can be used by default in the directory and a list of the most commonly used attributes, see the *Red Hat Directory Server 11 Configuration, Command, and File Reference*.

### B.4.1. Specifying Domain Entries

Directories often have at least one domain entry. Typically this is the first, or topmost, entry in the directory. The domain entry often corresponds to the DNS host and domain name for your directory. For example, if the Directory Server host is called **ldap.example.com**, then the domain entry for the directory is probably named **dc=ldap,dc=example,dc=com** or simply **dc=example,dc=com**.

The LDIF entry used to define a domain appears as follows:

```
dn: distinguished_name
objectClass: top
objectClass: domain
dc: domain_component_name
list_of_optional_attributes
...
...
```

The following is a sample domain entry in LDIF format:

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
description: Fictional example company
```

Each element of the LDIF-formatted domain entry is defined in [Table B.2, “LDIF Elements in Domain Entries”](#).

**Table B.2. LDIF Elements in Domain Entries**

LDIF Element	Description
dn: <i>distinguished_name</i>	<i>Required</i> . Specifies the distinguished name for the entry.
objectClass: top	<i>Required</i> . Specifies the <b>top</b> object class.

LDIF Element	Description
objectClass: domain	Specifies the <b>domain</b> object class. This line defines the entry as a domain or domain component. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes that can be used with this object class. -->
dc: <i>domain_component</i>	Attribute that specifies the domain's name. The server is typically configured during the initial setup to have a suffix or naming context in the form <b>dc=hostname,dc=domain,dc=toplevel</b> . For example, <b>dc=ldap,dc=example,dc=com</b> . The domain entry should use the leftmost <b>dc</b> value, such as <b>dc: ldap</b> . If the suffix were <b>dc=example,dc=com</b> , the <b>dc</b> value is <b>dc: example</b> . Do not create the entry for <b>dn: dc=com</b> unless the server has been configured to use that suffix.
<i>list_of_attributes</i>	Specifies the list of optional attributes to maintain for the entry. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes that can be used with this object class.

### B.4.2. Specifying Organizational Unit Entries

Organizational unit entries are often used to represent major branch points, or subdirectories, in the directory tree. They correspond to major, reasonably static entities within the enterprise, such as a subtree that contains people or a subtree that contains groups.

The organizational unit attribute that is contained in the entry may also represent a major organization within the company, such as marketing or engineering. However, this style is discouraged. Red Hat strongly encourages using a flat directory tree.

There is usually more than one organizational unit, or branch point, within a directory tree.

The LDIF that defines 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 example organizational unit
```

[Table B.3, “LDIF Elements in Organizational Unit Entries”](#) defines each element of the LDIF-formatted organizational unit entry.

**Table B.3. LDIF Elements in Organizational Unit Entries**

LDIF Element	Description
dn: <i>distinguished_name</i>	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 (\), such as <b>dn: ou=people,dc=example,dc=com</b> .
objectClass: top	<i>Required</i> . Specifies the <b>top</b> object class.
objectClass: organizationalUnit	Specifies the <b>organizationalUnit</b> object class. This line defines the entry as an <b>organizational unit</b> . See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes available for this object class.
ou: <i>organizational_unit_name</i>	Attribute that specifies the organizational unit's name.
<i>list_of_attributes</i>	Specifies the list of optional attributes to maintain for the entry. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes available for this object class.

### B.4.3. Specifying Organizational Person Entries

The majority of the entries in the 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: people
description: Fictional example person
telephoneNumber: 555-5557
userPassword: {SSHA}dkfljlk34r2kljdsfk9

```

Table B.4, “LDIF Elements in Person Entries” defines each aspect of the LDIF person entry.

**Table B.4. LDIF Elements in Person Entries**

LDIF Element	Description
dn: <i>distinguished_name</i>	<p><i>Required.</i> Specifies the distinguished name for the entry. For example, <b>dn: uid=bjensen,ou=people,dc=example,dc=com</b>. If there is a comma in the DN, the comma must be escaped with a backslash (\).</p>
objectClass: top	<p><i>Required.</i> Specifies the <b>top</b> object class.</p>
objectClass: person	<p>Specifies the <b>person</b> 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.</p>
objectClass: organizationalPerson	<p>Specifies the <b>organizationalPerson</b> object class. This object class specification should be included because some LDAP clients require it during search operations for an organizational person.</p>
objectClass: inetOrgPerson	<p>Specifies the <b>inetOrgPerson</b> object class. The <b>inetOrgPerson</b> object class is recommended for the creation of an organizational person entry because this object class includes the widest range of attributes. The <b>uid</b> attribute is required by this object class, and entries that contain this object class are named based on the value of the <b>uid</b> attribute. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes available for this object class.</p>
cn: <i>common_name</i>	<p>Specifies the person's common name, which is the full name commonly used by the person. For example, <b>cn: Bill Anderson</b>. At least one common name is required.</p>
sn: <i>surname</i>	<p>Specifies the person's surname, or last name. For example, <b>sn: Anderson</b>. A surname is required.</p>

LDIF Element	Description
<i>list_of_attributes</i>	Specifies the list of optional attributes to maintain for the entry. See the <a href="#">Red Hat Directory Server 11 Configuration, Command, and File Reference</a> for a list of the attributes available for this object class.

## B.5. DEFINING DIRECTORIES USING LDIF

The contents of an entire directory can be defined using LDIF. Using LDIF is an efficient method of directory creation when there are many entries to add to the directory.

To create a directory using LDIF:

1. Create an ASCII file containing the entries to add in LDIF format.

Make sure each entry is separated from the next by an empty line. Use just one line between entries, and make sure the first line of the file is not be blank, or else the **ldapmodify** utility will exit. For more information, see [Section B.4, "Specifying Directory Entries Using LDIF"](#).

2. Begin each file with the topmost, or root, entry in the database.

The root entry must represent the suffix or sub-suffix contained by the database. For example, if the database has the suffix **dc=example,dc=com**, the first entry in the 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 to create under that branch.

For example, to place an entry in a people and a group subtree, create the branch point for those subtrees before creating entries within those subtrees.



### NOTE

The LDIF file is read in order, so parent entries must be listed before the child entries.

4. Create the directory from the LDIF file using one of the following methods:

- o *Initializing the database through the web console.* Use this method if there is a small database to import (less than 10,000 entries). See [Section 6.1.3, "Importing Data Using the Web Console"](#).

**WARNING**

This method is destructive and will erase any existing data in the suffix.

- *ldif2db* or *ldif2db.pl* command-line utility. Use this method if there is a large database to import (more than 10,000 entries). See [Section 6.1.2.2, “Importing Data While the Server is Offline”](#).
  - **ldif2db** cannot be used if the server is running.
  - **ldif2db.pl** can only be used if the server is running.

**WARNING**

This method is destructive and will erase any existing data in the suffix.

- *ldapmodify* command-line utility with the *-a* parameter. Use this method if a new subtree is being added to an existing database or there is existing data in the suffix which should not be deleted. Unlike the other methods for creating the directory from an LDIF file, Directory Server must be running before a subtree can be added using **ldapmodify**. See [Section 3.3, “Adding an Entry”](#).

### Example B.1. LDIF File Example

This LDIF file contains one domain, two organizational units, and three organizational person entries:

```

dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
description: Fictional example domain

dn: ou=People,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People
description: Fictional example organizational unit
tel: 555-5559

dn: cn=June Rossi,ou=People,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,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,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups
description: Fictional example organizational unit

```

## B.6. STORING INFORMATION IN MULTIPLE LANGUAGES

If the directory contains a single language, it is not necessary to do anything special to add a new entry to the directory. However, if an organization is multinational, it may be necessary to store information in multiple languages so that users in different locales can view directory information in their own language.

When information in the directory is represented in multiple languages, the server associates language tags with attribute values. When a new entry is added, the attribute values used in the RDN (relative distinguished name, the naming attribute) must be provided without any language codes.

Multiple languages can be stored for a single attribute. In this case, 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 [Section D.2, "Supported Locales"](#).



### 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. The user is responsible for converting the data used in the LDIF to UTF-8. The **iconv** or **uconv** command provided by most operating systems can be used to convert data from the native character set into UTF-8.

For example, Example 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 does the following:

1. The administrator creates a file, **street.txt**, with the French street address value:

```
1 rue de l'Université
```

2. The file contents are then converted to UTF-8:

```
# iconv -t UTF-8 -o output.txt street.txt
```

3. The following LDIF entry is created using the UTF-8 value of the street address value for **streetAddress;lang-fr**.

```
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:: AasljdoaAJASI023909jaASJaonasd0ADS
preferredLanguage: fr
```

The double colons after the attribute name and subtype indicate that the value is binary base-64 encoded.

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 C. LDAP URLs

LDAP URLs identify the Red Hat Directory Server instance, similarly to the way site URLs identify a specific website or web page. There are three common times when the LDAP URL of the Directory Server instance is used:

- The LDAP URL is used to identify the specific Directory Server instance when the Directory Server is accessed using a web-based client.
- LDAP URLs are used to configure Directory Server referrals.
- LDAP URLs are used to configure access control instructions.



### NOTE

The LDAP URL format is described in RFC 4516, which is available at <http://www.ietf.org/rfc/rfc4516.txt>.

### C.1. COMPONENTS OF AN LDAP URL

`ldap[s]://hostname:port/base_dn?attributes?scope?filter`

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

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 TLS connections. [Table C.1, “LDAP URL Components”](#) lists the components of an LDAP URL.



### NOTE

The LDAP URL format is described in RFC 4516, which is available at <http://www.ietf.org/rfc/rfc4516.txt>.

**Table C.1. LDAP URL Components**

Component	Description
host name	Name, IPv4, or IPv6 address of the LDAP server. For example, <b>ldap.example.com</b> , <b>192.0.2.90</b> , or [2001:db8::1].
port	Port number of the LDAP server (for example, <b>696</b> ). If no port is specified, the standard LDAP port ( <b>389</b> ) or LDAPS port ( <b>636</b> ) is used.
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, <b>cn,mail,telephoneNumber</b> . If no attributes are specified in the URL, all attributes are returned.

Component	Description
scope	<p>The scope of the search, which can be one of these values:</p> <p><b>base</b> retrieves information only about the distinguished name (<i>base_dn</i>) specified in the URL.</p> <p><b>one</b> retrieves information about entries one level below the distinguished name (<i>base_dn</i>) specified in the URL. The base entry is not included in this scope.</p> <p><b>sub</b> retrieves information about entries at all levels below the distinguished name (<i>base_dn</i>) specified in the URL. The base entry is included in this scope.</p> <p>If no scope is specified, the server performs a <b>base</b> search.</p>
filter	Search filter to apply to entries within the specified scope of the search. If no filter is specified, the server uses the filter <b>(objectClass=*)</b> .

The attributes, scope, and filter components are identified by their positions in the URL. Even if no attributes are specified, the question marks still must be included to delimit 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 following LDAP 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.

## C.2. ESCAPING UNSAFE CHARACTERS

Any *unsafe* characters in the URL need to be escaped, or substituted with a special sequence of 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

Unsafe Character	Escape Characters
"	%22
#	%23
%	%25
{	%7b
}	%7d
	%7c
\	%5c
^	%5e
~	%7e
[	%5b
]	%5d
`	%60

### C.3. EXAMPLES OF LDAP URLs



#### NOTE

The LDAP URL format is described in RFC 4516, which is available at <http://www.ietf.org/rfc/rfc4516.txt>.

#### 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?cn,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.

# APPENDIX D. INTERNATIONALIZATION

Red Hat Directory Server allows users 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 languages they understand.

Directory Server supports all international character sets by default because directory data is stored in UTF-8. Further, Directory Server can use specified matching rules and collation orders based on language preferences in search operations.



## NOTE

ASCII characters are required for attribute and object class names.

## D.1. 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 defines four things:

- *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. For example, in some languages, the pipe (|) character is considered punctuation while in others it is considered alphabetic. In addition, it defines the mapping of upper-case to lower-case letters.
- *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 Czech 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 users as well as be presented in a way that users in a given region expect.

## D.2. SUPPORTED LOCALES

When performing directory operations that require that a locale be specified, such as a search operation, 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 two-character string for the country code, 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 for 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 to use. However, when setting up an international index, the OIDs must be used. For more information on indexing, see [Chapter 13, Managing Indexes](#).

For a list of language tags and OIDs supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

### D.3. 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 [Section 3.9, “Updating an Entry in an Internationalized Directory”](#). [Table D.1, “Supported Language Subtypes”](#) lists the supported language subtypes for Directory Server.

**Table D.1. Supported Language Subtypes**

Language Tag	Language
af	Afrikaans
be	Belarusian
bg	Bulgarian
ca	Catalan
cs	Czech
da	Danish
de	German
el	Greek
en	English
es	Spanish

Language Tag	Language
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
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sk	Slovak
sl	Slovenian
sq	Albanian
sr	Serbian

Language Tag	Language
sv	Swedish
tr	Turkish
uk	Ukrainian
zh	Chinese

## D.4. SEARCHING AN INTERNATIONALIZED DIRECTORY

When performing search operations, the Directory Server can 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 [Section D.2, "Supported Locales"](#).



### NOTE

An LDAPv3 search is required to perform internationalized searches. Therefore, do not set the LDAPv2 option on the call for **Idapsearch**.

This section focuses using matching rule filters to return international attribute values. For more information on general **Idapsearch** syntax, see [Section 14.2, "LDAP Search Filters"](#).

- [Section D.4.1, "Matching Rule Formats"](#)
- [Section D.4.2, "Supported Search Types"](#)
- [Section D.4.3, "International Search Examples"](#)

### D.4.1. Matching Rule Formats

The matching rule filters for internationalized searches can be represented in any several ways, and which one should be used is a matter of preference:

- As the OID of the collation order for the locale on which to base the search.
- As the language tag associated with the collation order on which to base the 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:

- [Section D.4.1.1, "Using an OID for the Matching Rule"](#)
- [Section D.4.1.2, "Using a Language Tag for the Matching Rule"](#)
- [Section D.4.1.3, "Using an OID and Suffix for the Matching Rule"](#)

- [Section D.4.1.4, “Using a Language Tag and Suffix for the Matching Rule”](#)

#### D.4.1.1. Using an OID for the Matching Rule

Each locale supported by the Directory Server has an associated collation order OID. For a list of OIDs supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

The collation order OID can be used 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**

#### D.4.1.2. Using a Language Tag for the Matching Rule

Each locale supported by the Directory Server has an associated language tag. For a list of language tags supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

The language tag can be used 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**

#### D.4.1.3. Using an OID and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, 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***



#### NOTE

This syntax is only supported by the **mozldap** utility and not by OpenLDAP utilities, such as **ldapsearch**.

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 OIDs supported by the Directory Server, see the [`/etc/dirsrv/config slapd-collations.conf`](#) file. For a list of relational operators and their equivalent suffixes, see [Table D.2, “Search Types, Operators, and Suffixes”](#).

#### D.4.1.4. Using a Language Tag and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, 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`



##### NOTE

This syntax is only supported by the **mozldap** utility and not by OpenLDAP utilities, such as **ldapsearch**.

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 language tags supported by the Directory Server, see the [`/etc/dirsrv/config slapd-collations.conf`](#) file. For a list of relational operators and their equivalent suffixes, see [Table D.2, “Search Types, Operators, and Suffixes”](#).

#### D.4.2. 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, either use the standard operators (=, >=, >, <, <=) in the value portion of the search string, or 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 D.2, “Search Types, Operators, and Suffixes”](#) summarizes each type of search, the operator, and the equivalent suffix.

**Table D.2. 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

### D.4.3. 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.

#### D.4.3.1. Less-Than Example

Performing a locale-specific search using the less-than operator (<), or suffix (.1) searches 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, any of the following matching rule filters would work:

```
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
```

#### D.4.3.2. Less-Than or Equal-to Example

Performing a locale-specific search using the less-than or equal-to operator (<=), or suffix (.2) searches 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, any of the following matching rule filters would work:

```
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
```

#### D.4.3.3. Equality Example

Performing a locale-specific search using the equal to operator (=), or suffix (.3) searches 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, any of the following matching rule filters would work:

```
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
```

#### D.4.3.4. Greater-Than or Equal-to Example

Performing a locale-specific search using the greater-than or equal-to operator (>=), or suffix (.4) searches 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, any of the following matching rule filters would work:

```
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
```

#### D.4.3.5. Greater-Than Example

Performing a locale-specific search using the greater-than operator (>), or suffix (.5) searches 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 **schránka4** in the Czech collation order, any of the following matching rule filters would work:

```
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:==schránka4  
...  
mailHost:cs.5:==schránka4
```

#### D.4.3.6. Substring Example

Performing an international substring search searches 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, any of the following matching rule filters would work:

```
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 the **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)
```

## D.5. 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 -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -W -b "dc=example,dc=com"
"sn:2.16.840.1.113730.3.3.2.7.1:=passin"

ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -W -b "dc=example,dc=com"
"sn:de:=passin"
```

However, the rules listed below will work (note the **.3** before the **passin** value):

```
# ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -W -b "dc=example,dc=com"
"sn:2.16.840.1.113730.3.3.2.7.1.3:=passin"

ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -W -b "dc=example,dc=com"
"sn:de.3:=passin"
```

## APPENDIX E. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Directory Server.

<b>Revision 11.4-1</b>	<b>Tue Nov 09 2021</b>	<b>Marc Muehlfeld</b>
Red Hat Directory Server 11.4 release of this guide.		
<b>Revision 11.3-1</b>	<b>Tue May 11 2021</b>	<b>Marc Muehlfeld</b>
Red Hat Directory Server 11.3 release of this guide.		
<b>Revision 11.2-1</b>	<b>Tue Nov 03 2020</b>	<b>Marc Muehlfeld</b>
Red Hat Directory Server 11.2 release of this guide.		
<b>Revision 11.1-1</b>	<b>Tue Apr 28 2020</b>	<b>Marc Muehlfeld</b>
Red Hat Directory Server 11.1 release of this guide.		
<b>Revision 11.0-1</b>	<b>Tue Nov 05 2019</b>	<b>Marc Muehlfeld</b>
Red Hat Directory Server 11.0 release of this guide.		