

Red Hat Directory Server 8.0

8.0

Administrator's Guide

ISBN:

Publication date:

Red Hat Directory Server 8.0: Administrator's Guide

Copyright © 2008 Red Hat, Inc.

Copyright © You need to override this in your local ent file Red Hat. This material may only be distributed subject to the terms and conditions set forth in the Open Publication License, V1.0 or later with the restrictions noted below (the latest version of the OPL is presently available at <http://www.opencontent.org/openpub/>).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

Red Hat and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

1801 Varsity Drive
Raleigh, NC 27606-2072
USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park, NC 27709
USA

Preface	xvii
1. Directory Server Overview	xvii
2. Example and Default References	xviii
3. Document Conventions	xviii
4. Related Information	xx
1. General Red Hat Directory Server Usage	1
1. Directory Server File Locations	1
2. LDAP Tool Locations	4
3. Starting and Stopping Servers	4
3.1. Starting and Stopping Directory Server from the Console	5
3.2. Starting and Stopping Directory Server from the Command Line	6
3.3. Starting and Stopping Administration Server	6
4. Starting the Directory Server Console	7
4.1. Logging into Directory Server	8
4.2. Changing Login Identity	8
4.3. Viewing the Current Console Bind DN	9
5. Changing Directory Server Port Numbers	9
6. Creating a New Directory Server Instance	11
7. Configuring the Directory Manager	12
2. Creating Directory Entries	15
1. Managing Entries from the Directory Console	15
1.1. Creating a Root Entry	15
1.2. Creating Directory Entries	16
1.3. Modifying Directory Entries	18
1.4. Deleting Directory Entries	23
2. Managing Entries from the Command-Line	24
2.1. Providing Input from the Command-Line	24
2.2. Creating a Root Entry from the Command-Line	25
2.3. Adding Entries Using LDIF	26
2.4. Adding and Modifying Entries Using ldapmodify	26
2.5. Deleting Entries Using ldapdelete	29
2.6. Using Special Characters	31
3. Tracking Modifications to Directory Entries	31
4. LDIF Update Statements	32
4.1. Adding an Entry Using LDIF	33
4.2. Renaming an Entry Using LDIF	35
4.3. Modifying an Entry Using LDIF	36
4.4. Deleting an Entry Using LDIF	40
4.5. Modifying an Entry in an Internationalized Directory	41
5. Maintaining Referential Integrity	41
5.1. How Referential Integrity Works	41
5.2. Using Referential Integrity with Replication	42
5.3. Enabling/Disabling Referential Integrity	43
5.4. Modifying the Update Interval	43
5.5. Modifying the Attribute List	44
3. Configuring Directory Databases	47

1. Creating and Maintaining Suffixes	47
1.1. Creating Suffixes	48
1.2. Maintaining Suffixes	54
2. Creating and Maintaining Databases	56
2.1. Creating Databases	56
2.2. Maintaining Directory Databases	61
2.3. Database Encryption	64
3. Creating and Maintaining Database Links	69
3.1. Configuring the Chaining Policy	69
3.2. Creating a New Database Link	75
3.3. Chaining Using SSL	85
3.4. Maintaining Database Links	85
3.5. Database Links and Access Control Evaluation	87
3.6. Advanced Feature: Tuning Database Link Performance	88
3.7. Advanced Feature: Configuring Cascading Chaining	92
4. Using Referrals	105
4.1. Starting the Server in Referral Mode	105
4.2. Setting Default Referrals	106
4.3. Creating Smart Referrals	107
4.4. Creating Suffix Referrals	109
4. Populating Directory Databases	113
1. Importing Data	113
1.1. Importing a Database from the Console	114
1.2. Initializing a Database from the Console	115
1.3. Importing from the Command-Line	116
2. Exporting Data	119
2.1. Exporting Directory Data to LDIF Using the Console	121
2.2. Exporting a Single Database to LDIF Using the Console	122
2.3. Exporting to LDIF from the Command-Line	122
3. Backing up and Restoring Data	124
3.1. Backing up All Databases	124
3.2. Backing up the dse.ldif Configuration File	126
3.3. Restoring All Databases	126
3.4. Restoring a Single Database	128
3.5. Restoring Databases That Include Replicated Entries	129
3.6. Restoring the dse.ldif Configuration File	129
5. Managing Entries with Roles, Class of Service, and Views	131
1. Using Roles	131
1.1. About Roles	131
1.2. Managing Roles Using the Console	133
1.3. Managing Roles Using the Command-Line	139
1.4. Using Roles Securely	142
2. Assigning Class of Service	143
2.1. About CoS	144
2.2. Managing CoS Using the Console	149
2.3. Managing CoS from the Command-Line	153
2.4. Creating Role-Based Attributes	160

2.5. Access Control and CoS	162
3. Using Views	162
3.1. Creating Views in the Console	163
3.2. Deleting Views from the Directory Server Console	164
3.3. Creating Views from the Command Line	164
3.4. Deleting Views from the Command Line	165
4. Using Groups	165
4.1. Managing Static Groups	165
4.2. Managing Dynamic Groups	167
6. Managing Access Control	169
1. Access Control Principles	169
1.1. ACI Structure	169
1.2. ACI Placement	170
1.3. ACI Evaluation	170
1.4. ACI Limitations	170
2. Default ACIs	171
3. Creating ACIs Manually	172
3.1. The ACI Syntax	173
3.2. Defining Targets	173
3.3. Defining Permissions	180
4. Bind Rules	184
4.1. Bind Rule Syntax	185
4.2. Defining User Access - userdn Keyword	186
4.3. Defining Group Access - groupdn Keyword	190
4.4. Defining Role Access - roledn Keyword	190
4.5. Defining Access Based on Value Matching	191
4.6. Defining Access from a Specific IP Address	196
4.7. Defining Access from a Specific Domain	197
4.8. Defining Access at a Specific Time of Day or Day of Week	198
4.9. Defining Access Based on Authentication Method	199
4.10. Using Boolean Bind Rules	201
5. Creating ACIs from the Console	202
5.1. Displaying the Access Control Editor	203
5.2. Creating a New ACI	204
5.3. Editing an ACI	209
5.4. Deleting an ACI	210
6. Viewing ACIs	210
7. Get Effective Rights Control	211
7.1. Using Get Effective Rights from the Command-Line	212
7.2. Using Get Effective Rights from the Console	215
7.3. Get Effective Rights Return Codes	215
8. Logging Access Control Information	216
9. Access Control Usage Examples	216
9.1. Granting Anonymous Access	217
9.2. Granting Write Access to Personal Entries	219
9.3. Restricting Access to Key Roles	222
9.4. Granting a Group Full Access to a Suffix	224

9.5. Granting Rights to Add and Delete Group Entries	225
9.6. Granting Conditional Access to a Group or Role	227
9.7. Denying Access	229
9.8. Setting a Target Using Filtering	232
9.9. Allowing Users to Add or Remove Themselves from a Group	232
9.10. Defining Permissions for DNs That Contain a Comma	234
9.11. Proxied Authorization ACI Example	234
10. Advanced Access Control: Using Macro ACIs	235
10.1. Macro ACI Example	235
10.2. Macro ACI Syntax	237
11. Access Control and Replication	241
12. Compatibility with Earlier Releases	241
7. Managing User Accounts and Passwords	243
1. Managing the Password Policy	243
1.1. Configuring the Password Policy	243
1.2. Setting User Passwords	255
1.3. Password Change Extended Operation	255
1.4. Configuring the Account Lockout Policy	257
1.5. Managing the Password Policy in a Replicated Environment	260
1.6. Synchronizing Passwords	260
2. Inactivating Users and Roles	261
2.1. Inactivating User and Roles Using the Console	262
2.2. Inactivating User and Roles Using the Command-Line	262
2.3. Activating User and Roles Using the Console	263
2.4. Activating User and Roles Using the Command-Line	263
3. Setting Resource Limits Based on the Bind DN	264
3.1. Setting Resource Limits Using the Console	264
3.2. Setting Resource Limits Using the Command-Line	265
8. Managing Replication	267
1. Replication Overview	267
1.1. What Directory Units Are Replicated	267
1.2. Read-Write and Read-Only Replicas	267
1.3. Suppliers and Consumers	268
1.4. Changelog	268
1.5. Replication Identity	268
1.6. Replication Agreement	269
1.7. Compatibility with Earlier Versions of Directory Server	269
2. Replication Scenarios	270
2.1. Single-Master Replication	270
2.2. Multi-Master Replication	271
2.3. Cascading Replication	274
3. Creating the Supplier Bind DN Entry	275
4. Configuring Single-Master Replication	276
4.1. Configuring the Read-Write Replica on the Supplier Server	277
4.2. Configuring the Read-Only Replica on the Consumer	278
4.3. Create the Replication Agreement	280
5. Configuring Multi-Master Replication	285

5.1. Configuring the Read-Write Replicas on the Supplier Servers	286
5.2. Configuring the Read-Only Replicas on the Consumer Servers	289
5.3. Setting up the Replication Agreements	291
5.4. Preventing Monopolization of the Consumer in Multi-Master Replication	297
6. Configuring Cascading Replication	298
6.1. Configuring the Read-Write Replica on the Supplier Server	299
6.2. Configuring the Read-Only Replica on the Consumer Server	300
6.3. Configuring the Read-Only Replica on the Hub	302
6.4. Setting up the Replication Agreements	305
7. Configuring Replication from the Command Line	311
7.1. Configuring Suppliers from the Command Line	311
7.2. Configuring Consumers from the Command Line	315
7.3. Configuring Hubs from the Command Line	316
7.4. Configuring Replication Agreements from the Command Line	317
7.5. Initializing Consumers Online from the Command Line	321
8. Making a Replica Updatable	322
9. Deleting the Changelog	322
9.1. Removing the Changelog	323
9.2. Moving the Changelog to a New Location	323
10. Initializing Consumers	323
10.1. When to Initialize a Consumer	324
10.2. Online Consumer Initialization Using the Console	324
10.3. Initializing Consumers Online Using the Command Line	325
10.4. Manual Consumer Initialization Using the Command Line	326
10.5. Filesystem Replica Initialization	327
11. Forcing Replication Updates	329
11.1. Forcing Replication Updates from the Console	329
11.2. Forcing Replication Updates from the Command-Line	330
12. Replicating Account Lockout Attributes	331
13. Replication over SSL	332
14. Replicating o=NetscapeRoot for Administration Server Failover	333
15. Replication with Earlier Releases	335
16. Using the Retro Changelog Plug-in	336
16.1. Enabling the Retro Changelog Plug-in	337
16.2. Trimming the Retro Changelog	338
16.3. Searching and Modifying the Retro Changelog	339
16.4. Retro Changelog and the Access Control Policy	339
17. Monitoring Replication Status	339
17.1. Monitoring Replication Status from the Directory Server Console	339
17.2. Monitoring Replication Status from Administration Express	340
18. Solving Common Replication Conflicts	342
18.1. Solving Naming Conflicts	343
18.2. Solving Orphan Entry Conflicts	346
18.3. Solving Potential Interoperability Problems	346
19. Troubleshooting Replication-Related Problems	347
9. Extending the Directory Schema	353
1. Overview of Extending Schema	353

2. Managing Attributes	353
2.1. Viewing Attributes	353
2.2. Creating Attributes	355
2.3. Editing Attributes	356
2.4. Deleting Attributes	356
3. Managing Object Classes	357
3.1. Viewing Object Classes	357
3.2. Creating Object Classes	359
3.3. Editing Object Classes	360
3.4. Deleting Object Classes	361
4. Turning Schema Checking On and Off	362
10. Managing Indexes	363
1. About Indexes	363
1.1. About Index Types	363
1.2. About Default, System, and Standard Indexes	364
1.3. Overview of the Searching Algorithm	367
1.4. Approximate Searches	369
1.5. Balancing the Benefits of Indexing	370
2. Creating Indexes	371
2.1. Creating Indexes from the Server Console	372
2.2. Creating Indexes from the Command-Line	373
2.3. Creating Browsing Indexes from the Server Console	377
2.4. Creating Browsing Indexes from the Command-Line	377
3. Deleting Indexes	381
3.1. Deleting Indexes from the Server Console	382
3.2. Deleting Indexes from the Command-Line	383
3.3. Deleting Browsing Indexes from the Server Console	385
3.4. Deleting Browsing Indexes from the Command-Line	385
4. Managing Indexes	388
4.1. Indexing Performance	388
4.2. Search Performance	389
4.3. Backwards Compatibility and Migration	390
5. Attribute Name Quick Reference Table	390
11. Managing SSL	393
1. Introduction to SSL in the Directory Server	393
1.1. Enabling SSL: Summary of Steps	393
1.2. Command-Line Functions for Start TLS	394
2. Obtaining and Installing Server Certificates	395
2.1. Step 1: Generate a Certificate Request	396
2.2. Step 2: Send the Certificate Request	399
2.3. Step 3: Install the Certificate	400
2.4. Step 4: Trust the Certificate Authority	401
2.5. Step 5: Confirm That The New Certificates Are Installed	402
3. Using certutil	402
3.1. Creating Directory Server Certificates through the Command Line	402
3.2. certutil Usage	405
4. Starting the Server with SSL Enabled	405

4.1. Enabling SSL Only in the Directory Server	406
4.2. Enabling SSL in the Directory Server, Administration Server, and Console	408
4.3. Creating a Password File for the Directory Server	410
4.4. Creating a Password File for the Administration Server	411
5. Setting Security Preferences	412
5.1. Available Ciphers	412
5.2. Selecting the Encryption Cipher	414
6. Using Certificate-Based Authentication	415
6.1. Setting up Certificate-Based Authentication	416
6.2. Allowing/Requiring Client Authentication	416
7. Configuring LDAP Clients to Use SSL	417
12. Managing SASL	421
1. Authentication Mechanisms	421
2. SASL Identity Mapping	422
3. Configuring SASL Identity Mapping from the Console	424
4. Configuring SASL Identity Mapping from the Command-Line	426
5. Configuring Kerberos	426
5.1. Realms	427
5.2. Configuring the KDC Server	427
5.3. Example: Configuring an Example KDC Server	428
5.4. Configuring SASL Authentication at Directory Server Startup	429
13. Monitoring Server and Database Activity	431
1. Viewing and Configuring Log Files	431
1.1. Defining a Log File Rotation Policy	431
1.2. Defining a Log File Deletion Policy	433
1.3. Access Log	433
1.4. Error Log	435
1.5. Audit Log	437
2. Manual Log File Rotation	438
3. Monitoring Server Activity	438
3.1. Monitoring the Server from the Directory Server Console	439
3.2. Monitoring the Directory Server from the Command Line	443
4. Monitoring Database Activity	445
4.1. Monitoring Database Activity from the Directory Server Console	445
4.2. Monitoring Databases from the Command Line	448
5. Monitoring Database Link Activity	451
14. Monitoring Directory Server Using SNMP	453
1. About SNMP	453
2. Configuring the Master Agent	454
3. Configuring the Subagent	454
3.1. Subagent Configuration File	454
3.2. Starting the Subagent	455
3.3. Testing the Subagent	456
4. Configuring SNMP Traps	456
5. Configuring the Directory Server for SNMP	457
6. Using the Management Information Base	457

6.1. Operations Table	458
6.2. Entries Table	460
6.3. Entity Table	460
6.4. Interaction Table	461
15. Tuning Directory Server Performance	463
1. Tuning Server Performance	463
2. Tuning Database Performance	464
2.1. Optimizing Search Performance	464
2.2. Tuning Transaction Logging	466
2.3. Changing the Location of the Database Transaction Log	466
2.4. Changing the Database Checkpoint Interval	467
2.5. Disabling Durable Transactions	468
2.6. Specifying Transaction Batching	468
3. Miscellaneous Tuning Tips	469
3.1. Avoid Creating Entries Under the cn=config Entry in the dse.ldif File	469
16. Administering Directory Server Plug-ins	471
1. Server Plug-in Functionality Reference	471
1.1. 7-Bit Check Plug-in	471
1.2. ACL Plug-in	471
1.3. ACL Preoperation Plug-in	472
1.4. Binary Syntax Plug-in	472
1.5. Boolean Syntax Plug-in	473
1.6. Case Exact String Syntax Plug-in	473
1.7. Case Ignore String Syntax Plug-in	474
1.8. Chaining Database Plug-in	474
1.9. Class of Service Plug-in	475
1.10. Country String Syntax Plug-in	475
1.11. Distinguished Name Syntax Plug-in	476
1.12. Generalized Time Syntax Plug-in	476
1.13. Integer Syntax Plug-in	477
1.14. Internationalization Plug-in	477
1.15. Idbm Database Plug-in	478
1.16. Legacy Replication Plug-in	478
1.17. Multi-Master Replication Plug-in	479
1.18. Octet String Syntax Plug-in	479
1.19. CLEAR Password Storage Plug-in	480
1.20. CRYPT Password Storage Plug-in	480
1.21. NS-MTA-MD5 Password Storage Plug-in	481
1.22. SHA Password Storage Plug-in	482
1.23. SSHA Password Storage Plug-in	482
1.24. Postal Address String Syntax Plug-in	483
1.25. PTA Plug-in	483
1.26. Referential Integrity Postoperation Plug-in	484
1.27. Retro Changelog Plug-in	485
1.28. Roles Plug-in	486
1.29. Space Insensitive String Syntax Plug-in	486
1.30. State Change Plug-in	487

1.31. Telephone Syntax Plug-in	488
1.32. UID Uniqueness Plug-in	488
1.33. URI Plug-in	489
2. Enabling and Disabling Plug-ins	490
17. Using the Pass-through Authentication Plug-in	491
1. How Directory Server Uses PTA	491
2. PTA Plug-in Syntax	492
3. Configuring the PTA Plug-in	495
3.1. Turning the Plug-in On or Off	496
3.2. Configuring the Servers to Use a Secure Connection	496
3.3. Specifying the Authenticating Directory Server	496
3.4. Specifying the Pass-through Subtree	497
3.5. Configuring the Optional Parameters	498
4. PTA Plug-in Syntax Examples	499
4.1. Specifying One Authenticating Directory Server and One Subtree	499
4.2. Specifying Multiple Authenticating Directory Servers	500
4.3. Specifying One Authenticating Directory Server and Multiple Subtrees	500
4.4. Using Non-Default Parameter Values	500
4.5. Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers	501
18. Using the Attribute Uniqueness Plug-in	503
1. Overview of the Attribute Uniqueness Plug-in	503
2. Attribute Uniqueness Plug-in Syntax	504
3. Creating an Instance of the Attribute Uniqueness Plug-in	506
4. Configuring Attribute Uniqueness Plug-ins	507
4.1. Viewing Plug-in Configuration Information	507
4.2. Configuring Attribute Uniqueness Plug-ins from the Directory Server Console	508
4.3. Configuring Attribute Uniqueness Plug-ins from the Command-Line	509
5. Attribute Uniqueness Plug-in Syntax Examples	511
5.1. Specifying One Attribute and One Subtree	511
5.2. Specifying One Attribute and Multiple Subtrees	511
6. Replication and the Attribute Uniqueness Plug-in	512
6.1. Simple Replication Scenario	512
6.2. Multi-Master Replication Scenario	513
19. Synchronizing Red Hat Directory Server with Microsoft Active Directory	515
1. About Windows Sync	515
2. Configuring Windows Sync	518
2.1. Step 1: Configure SSL on Directory Server	518
2.2. Step 2: Configure the Active Directory Domain	519
2.3. Step 3: Select or Create the Sync Identity	520
2.4. Step 4: Install and Configure the Password Sync Service	521
2.5. Step 5: Configure the Directory Server Database for Synchronization	524
2.6. Step 6: Create the Synchronization Agreement	525
2.7. Step 7: Begin Synchronization	527
3. Using Windows Sync	527
3.1. Synchronizing Users	528

3.2. Synchronizing Groups	530
3.3. Deleting Entries	531
3.4. Resurrecting Entries	532
3.5. Manually Updating and Resynchronizing Entries	532
3.6. Checking Synchronization Status	533
3.7. Modifying the Sync Agreement	533
4. Schema Differences	534
4.1. Password Policies	534
4.2. Groups	534
4.3. Values for street and streetAddress	534
4.4. Constraints on the initials attribute	535
5. Password Sync Service	535
5.1. Modifying Password Sync	535
5.2. Starting and Stopping the Password Sync Service	535
5.3. Uninstalling Password Sync Service	536
6. Troubleshooting	536
A. LDAP Data Interchange Format	539
1. About the LDIF File Format	539
2. Continuing Lines in LDIF	540
3. Representing Binary Data	541
3.1. Standard LDIF Notation	541
3.2. Base-64 Encoding	541
4. Specifying Directory Entries Using LDIF	542
4.1. Specifying Domain Entries	542
4.2. Specifying Organizational Unit Entries	544
4.3. Specifying Organizational Person Entries	545
5. Defining Directories Using LDIF	546
5.1. LDIF File Example	548
6. Storing Information in Multiple Languages	549
B. Finding Directory Entries	551
1. Finding Entries Using the Directory Server Console	551
2. Using Idapsearch	552
2.1. Using Special Characters	553
2.2. Idapsearch Command-Line Format	553
2.3. Commonly Used Idapsearch Options	554
2.4. Idapsearch Examples	556
3. LDAP Search Filters	559
3.1. Search Filter Syntax	560
4. Searching an Internationalized Directory	563
4.1. Matching Rule Filter Syntax	564
4.2. Supported Search Types	567
4.3. International Search Examples	568
C. LDAP URLs	571
1. Components of an LDAP URL	571
2. Escaping Unsafe Characters	573
3. Examples of LDAP URLs	573
D. Internationalization	577

1. About Locales	577
2. Identifying Supported Locales	578
3. Supported Language Subtypes	580
4. Troubleshooting Matching Rules	581
Glossary	583
Index	601

Preface

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

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

1. Directory Server Overview

Directory Server provides the following key features:

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

The major components of Directory Server include the following:

- An LDAP server — The LDAP v3-compliant network daemon.
- Directory Server Console — A graphical management console that dramatically reduces the

effort of setting up and maintaining your directory service.

- **SNMP agent** — Can monitor the Directory Server using the Simple Network Management Protocol (SNMP).
- **Directory Gateway** — A web application which allows users to search for information in the Directory Server, in addition to providing self service access to their own information, including password changes, to reduce user support costs.
- **Org Chart** — A web application which shows a graphical view of the structure of your organization.

2. Example and Default References

There are differences between the command, directory, and file locations in Red Hat Enterprise Linux, Sun Solaris, and HP-UX Directory Server installations. Locations for other platforms are listed in [Section 1, “Directory Server File Locations”](#). These differences impact the documentation in two ways:

- The file locations used in the examples or referenced in the procedures are the default locations on Red Hat Enterprise Linux.
- The default commands used in the examples are also the default commands on Red Hat Enterprise Linux.

There is another important consideration with the Directory Server tools. The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/dir/mozldap` directory on Red Hat Enterprise Linux (directories for other platforms are listed in [Chapter 1, General Red Hat Directory Server Usage](#)).

However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP in the `/usr/bin` directory. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL, which OpenLDAP tools use by default.

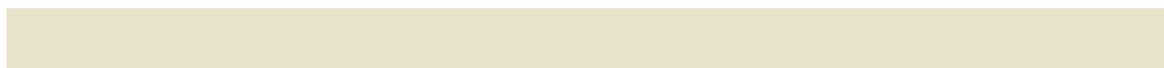
3. Document Conventions

Certain words in this manual are represented in different fonts, styles, and weights. This highlighting indicates that the word is part of a specific category. The categories include the following:

Courier font

Courier font represents commands, file names and paths, and prompts.

When shown as below, it indicates computer output:



Desktop	about.html	logs	paulwesterberg.png
Mail	backupfiles	mail	reports

Courier font

Bold Courier font represents text that you are to type, such as: `service jonas start`

If you have to run a command as root, the root prompt (`#`) precedes the command:

```
# gconftool-2
```

Courier font

Italic Courier font represents a variable, such as an installation directory:

```
install_dir/bin/
```

font

Bold font represents **application programs** and **text found on a graphical interface**.

When shown like this: **OK**, it indicates a button on a graphical application interface.

Additionally, the manual uses different strategies to draw your attention to pieces of information. In order of how critical the information is to you, these items are marked as follows:

**Note**

A note is typically information that you need to understand the behavior of the system.

**Tip**

A tip is typically an alternative way of performing a task.

**Important**

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



Caution

A caution indicates an act that would violate your support agreement, such as recompiling the kernel.



Warning

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

4. Related Information

This manual describes how to administer the Directory Server and its contents. The instructions for installing the various Directory Server components are contained in the *Red Hat Directory Server Installation Guide*.

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

- *Red Hat Directory Server Release Notes* - Contains important information on new features, fixed bugs, known issues and workarounds, and other important deployment information for this specific version of Directory Server.
- *Red Hat Directory Server Configuration, Command, and File Reference* - Provides reference information on the command-line scripts, configuration attributes, and log files shipped with Directory Server.
- *Red Hat Directory Server Installation Guide* - Contains procedures for installing your Directory Server as well as procedures for migrating from a previous installation of Directory Server.

For the latest information about Directory Server, including current release notes, complete product documentation, technical notes, and deployment information, see the Red Hat Directory Server documentation site at <http://www.redhat.com/docs/manuals/dir-server/>.

General Red Hat Directory Server Usage

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

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

Directory Server 8.0 is comprised of several components, which work in tandem:

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

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

1. Directory Server File Locations

Red Hat Directory Server 8.0 conforms to the Filesystem Hierarchy Standards. For more information on FHS, see the FHS homepage, <http://www.pathname.com/fhs/>. The files and directories installed with Directory Server are listed in the tables below for each supported platform.

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

The Administration Server directories are named the same as the Directory Server directories,

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

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

Table 1.1. Red Hat Enterprise Linux 4 and 5 (x86)

File or Directory	Location
Log files	<code>/var/log/dirsrv/slapd-<i>instance</i></code>
Configuration files	<code>/etc/dirsrv/slapd-<i>instance</i></code>
Instance directory	<code>/usr/lib64/dirsrv/slapd-<i>instance</i></code>
Database files	<code>/var/lib/dirsrv/slapd-<i>instance</i></code>
Runtime files	<code>/var/lock/dirsrv/slapd-<i>instance</i></code> <code>/var/run/dirsrv/slapd-<i>instance</i></code>
Initscripts	<code>/etc/rc.d/init.d/dirsrv</code> and <code>/etc/sysconfig/dirsrv</code> <code>/etc/rc.d/init.d/dirsrv-admin</code> and <code>/etc/sysconfig/dirsrv-admin</code>
Tools	<code>/usr/bin/</code> <code>/usr/sbin/</code>

File or Directory	Location
-------------------	----------

Table 1.2. Red Hat Enterprise Linux 4 and 5 (x86_64)

File or Directory	Location
Log files	/var/log/dirsrv/slapd- <i>instance</i>
Configuration files	/etc/dirsrv/slapd- <i>instance</i>
Instance directory	/usr/lib/sparc9/dirsrv/slapd- <i>instance</i>
Database files	/var/lib/dirsrv/slapd- <i>instance</i>
Runtime files	/var/lock/dirsrv/slapd- <i>instance</i> /var/run/dirsrv/slapd- <i>instance</i>
Initscripts	/etc/rc.d/init.d/dirsrv and /etc/default/dirsrv /etc/rc.d/init.d/dirsrv-admin and /etc/default/dirsrv-admin
Tools	/usr/bin/ /usr/sbin/

Table 1.3. Sun Solaris 9 (sparc)

File or Directory	Location
Log files	/var/opt/log/dirsrv/slapd- <i>instance</i>
Configuration files	/etc/opt/dirsrv/slapd- <i>instance</i>
Instance directory	/opt/dirsrv/slapd- <i>instance</i>
Database files	/var/opt/dirsrv/slapd- <i>instance</i>
Runtime files	/var/opt/dirsrv/ <i>instance</i>
Binaries	/opt/dirsrv/bin/ /opt/dirsrv/sbin/
Libraries	/opt/dirsrv/lib/

Table 1.4. HP-UX 11i (IA64)

2. LDAP Tool Locations

Red Hat Directory Server uses Mozilla LDAP tools — such as `ldapsearch`, `ldapmodify`, and `ldapdelete` — for command-line operations. The MozLDAP tools are installed with Directory Server.

Platform	Directory Location
Red Hat Enterprise Linux 4 i386	<code>/usr/lib/mozldap6</code>
Red Hat Enterprise Linux 4 x86_64	<code>/usr/lib64/mozldap6</code>
Red Hat Enterprise Linux 5 i386	<code>/usr/lib/mozldap</code>
Red Hat Enterprise Linux 5 x86_64	<code>/usr/lib64/mozldap</code>
Sun Solaris	<code>/usr/lib/sparcv9/mozldap</code>
HP-UX	<code>/opt/dirsrv/bin</code>

For all Red Hat Directory Server guides and documentation, the LDAP tools used in the examples, such as `ldapsearch` and `ldapmodify`, are the Mozilla LDAP tools. For most Linux systems, OpenLDAP tools are already installed in the `/usr/bin/` directory. These OpenLDAP tools are not supported for Directory Server operations. For the best results with the Directory Server, make sure the path to the Mozilla LDAP tools comes first in the `PATH` or use the full path and file name for every LDAP operation.

However, these OpenLDAP tools can be used for Directory Server operations with certain cautions:

- The output of the other tools may be different, so it may not look like the examples in the documentation.
- The OpenLDAP tools require a `-x` argument to disable SASL so that it can be used for a simple bind, meaning the `-D` and `-w` arguments or an anonymous bind.
- The OpenLDAP tools' arguments for using TLS/SSL and SASL are quite different than the Mozilla LDAP arguments. See the OpenLDAP documentation for instructions on those arguments.

3. Starting and Stopping Servers

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

- If the Directory Server has SSL enabled, you cannot restart the server from the Console; you must use the command-line. It is possible to restart without being prompted for a password; see [Section 4.3, “Creating a Password File for the Directory Server”](#) for more information.

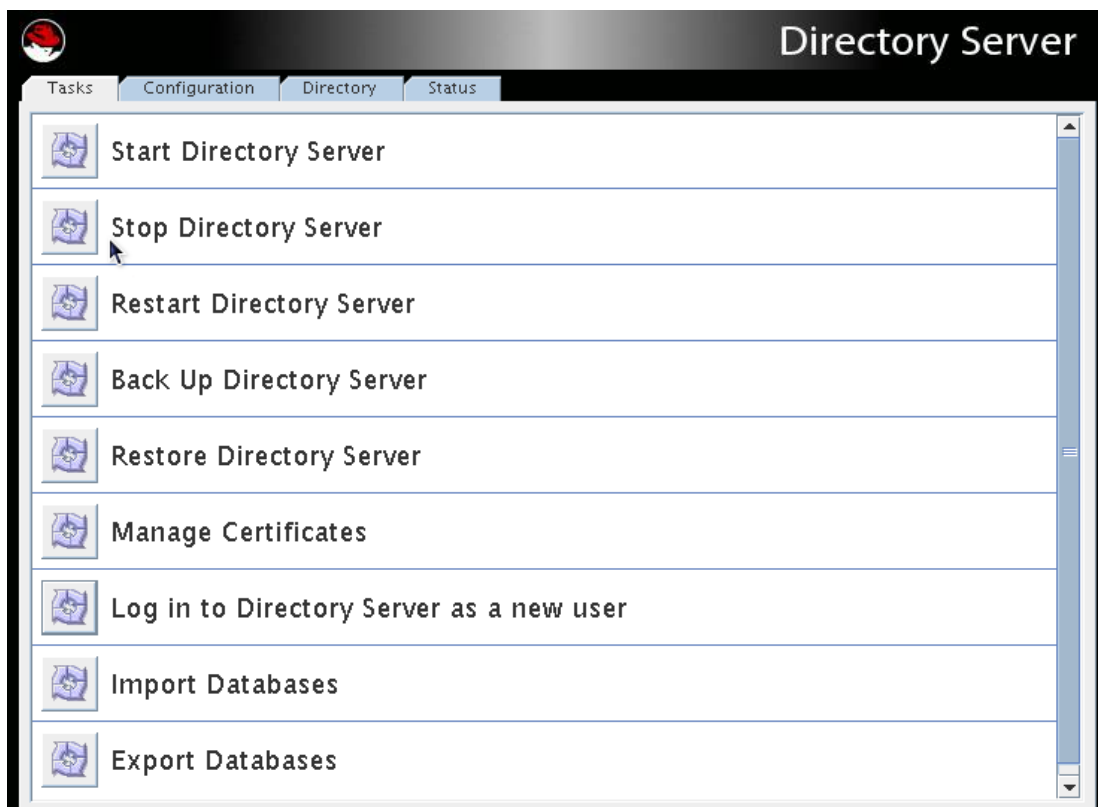
- Rebooting the host system can automatically start the `ns-slapd` process. The directory provides startup or run command (`rc`) scripts. On Red Hat Enterprise Linux, use the `chkconfig` command to enable the Directory Server and Administration Server to start on boot. On Solaris, the commands are already set up in the `/etc/rc.d` directories to start up the servers at boot time. For HP-UX, check the operating system documentation for details on adding these scripts.
- If the Directory Server shuts down due to a full disk, subsequent restart of the server may take a very long time, even more than an hour. Ensure that the machine on which you install the server has adequate disk space and that the machine is configured appropriately to handle large files. For more information on setting these parameters, the system requirements described in the *Directory Server Installation Guide*.

3.1. Starting and Stopping Directory Server from the Console

1. Start the Directory Server Console.

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

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



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

3.2. Starting and Stopping Directory Server from the Command Line

There are two ways to start, stop, or restart the Directory Server:

- There are scripts in the instance directories. For example:

```
/usr/lib/dirsrv/slapd-instance/start-slapd
/usr/lib/dirsrv/slapd-instance/restart-slapd
/usr/lib/dirsrv/slapd-instance/stop-slapd
```

- The Directory Server service can also be stopped and started using system tools on Red Hat Enterprise Linux and Solaris. For example, Linux uses the `service` tool:

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



NOTE

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

Solaris uses `/etc/init.d`:

```
/etc/init.d/dirsrv {start|stop|restart} instance
```

The Directory Server instance name can be specific in both the `start|stop|restart-slapd` and system scripts. If an instance name is not given, the start or stop operation applies to all instances on the machine.

3.3. Starting and Stopping Administration Server

There are two ways to start, stop, or restart the Administration Server:

- There are scripts in the `/usr/sbin` directory.

```
/usr/sbin/start|stop|restart-ds-admin
```

- The Administration Server service can also be stopped and started using system tools on Red Hat Enterprise Linux and Solaris. For example, on Red Hat Enterprise Linux, the command is `service`:

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



NOTE

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

On Solaris, the service is `init.d`:

```
/etc/init.d/dirsrv-admin {start|stop|restart}
```

4. Starting the Directory Server Console

There is a simple script to launch the Directory Server Console. On Red Hat Enterprise Linux and Solaris, run the following:

```
/usr/bin/redhat-idm-console
```

HP-UX has a different location for the script:

```
/opt/dirsrv/bin/redhat-idm-console
```



NOTE

Make sure that the correct JRE — the program called `java` — is set in the `PATH` before launching the Console. Run the following to see if the Java program is in the `PATH` and to get the version and vendor information:

```
java -version
```

When the login screen opens, you are prompted for the username, password, and Administration Server location. It is possible to send the Administration Server URL and port with the start script. For example:

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

The `a` option is a convenience, particularly if you are logging into a Directory Server for the first time. On subsequent logins, the URL is saved. If you do not pass the Administration Server port number with the `redhat-idm-console` command, then you are prompted for it at the Console login screen.

4.1. Logging into Directory Server

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

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



4.2. Changing Login Identity

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

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



3. A login dialog box appears.



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

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

4.3. Viewing the Current Console Bind DN

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



Figure 1.1. Viewing the Bind DN

5. Changing Directory Server Port Numbers

The standard and secure LDAP port numbers used by Directory Server can be changed through the Directory Server Console or by changing the value of the `nsslapd-port` or

`nsslapd-secureport` attribute under the `cn=config` entry in the `dse.ldif`.



NOTE

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

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

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

To modify a Directory Server LDAP or LDAPS port for either a user or a configuration directory, do the following:

1. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
2. Select the **Settings** tab in the right pane.
3. Enter the port number for the server to use for non-SSL communications in the **Port** field. The default value is 389.
4. Enter the port number for the server to use for SSL communications in the **Encrypted Port** field.

The encrypted port number must not be the same port number used for normal LDAP communications. The default value is 636.

5. Click **Save**.
6. The Console returns a warning, *You are about to change the port number for the Configuration Directory. This will affect all Administration Servers that use this directory and you'll need to update them with the new port number. Are you sure you want to change the port number?* Click **Yes**.
7. Then a dialog appears, reading that the changes will not take effect until the server is restarted. Click **OK**.



NOTE

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

8. Open the Administration Server Console.
9. In the **Configuration** tab, select the **Configuration DS** tab.
10. In the **LDAP Port** field, type in the new LDAP port number for your Directory Server instance.
11. Check the **Secure Connection** box if this is a secure port.



NOTE

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

12. In the **Tasks** tab of the Directory Server Console, click **Restart Directory Server**. A dialog to confirm that you want to restart the server. Click **Yes**.
13. Open the **Configuration DS** tab of the Administration Server Console and select **Save**.

A dialog will appear, reading *The Directory Server setting has been modified. You must shutdown and restart your Administration Server and all the servers in the Server Group for the changes to take effect*. Click **OK**.
14. In the **Tasks** tab of the Administration Server Console, click **Restart Admin Server**. A dialog opens reading that the Administration Server has been successfully restarted. Click **Close**.



NOTE

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

6. Creating a New Directory Server Instance

Additional instances can be created through the Directory Server Console or using the `setup-ds.pl` script. For information on using the `setup-ds.pl` script, see the *Directory Server*

Installation Guide. To create an instance using the Directory Server Console, do the following:

1. In the Red Hat Console window, select **Server Group** in the navigation tree, and then right-click.
2. From the pop-up menu, select **Create Instance** and then **Directory Server**.

The **Create New Instance** dialog box is displayed.

3. Enter a unique identifier for the server in the **Server Identifier** field.



NOTE

This name must only have alphanumeric characters, a dash (-), or an underscore (_).

4. Enter the a port number for LDAP communications in the **Network port** field.
5. Enter the suffix managed by this new instance of the directory in the **Base Suffix** field.
6. Enter a DN for the Directory Manager in the **Root DN** field.

For information on the role and privileges of the Directory Manager entry, refer to [Section 7, “Configuring the Directory Manager”](#).

7. Enter the password for this user in the **Password for Root DN** field, and confirm it.
8. Enter the user ID for the Directory Server daemon in the **Server Runtime User ID** field.
9. Click **OK**.

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

7. Configuring the Directory Manager

The Directory Manager is the privileged database administrator, comparable to the `root` user in UNIX. Access control does not apply to the Directory Manager entry; likewise, limits on searches and other operations do not apply. The Directory Manager entry is created during installation; the default DN is `cn=Directory Manager`. The password for this user is defined in the `nsslapd-rootdn` attribute.

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

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

If you are already logged in to the Console, change the bind DN, as described in [Section 4.2, “Changing Login Identity”](#).

2. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
3. Select the **Manager** tab in the right pane.
4. Enter the new distinguished name for the Directory Manager in the **Root DN** field.

The default value is `cn=Directory Manager`.

5. From the **Manager Password Encryption** pull-down menu, select the storage scheme you want the server to use to store the password for Directory Manager.
6. Enter the new password, and confirm it.
7. Click **Save**.

Creating Directory Entries

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

Entries stored in Active Directory can be added to the Directory Server through Windows Sync; see [Chapter 19, Synchronizing Red Hat Directory Server with Microsoft Active Directory](#) for more information on adding or modifying synchronized entries through Windows User Sync.

1. Managing Entries from the Directory Console

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

To add several entries simultaneously, use the command-line utilities described in [Section 1, “Managing Entries from the Directory Console”](#).

- [Section 1.1, “Creating a Root Entry”](#)
- [Section 1.2, “Creating Directory Entries”](#)
- [Section 1.3, “Modifying Directory Entries”](#)
- [Section 1.4, “Deleting Directory Entries”](#)



NOTE

You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for your directory, see [Chapter 6, Managing Access Control](#).

1.1. Creating a Root Entry

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

To create a root entry for a database, do the following:



NOTE

For information on starting the Directory Server Console, see [Section 4, “Starting the Directory Server Console”](#).

1. In the Directory Server Console, select the **Configuration** tab.
2. Create a new database, as explained in [Section 2, “Creating and Maintaining Databases”](#).
3. In the **Directory** tab, right-click the top object representing the Directory Server, and choose **New Root Object**.

The secondary menu under **New Root Object** displays a list of suffixes that do not have a corresponding entry.

4. Choose the suffix corresponding to the entry to create.

The **New Object** window opens.

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

The object class you select must contain the attribute you used to name the suffix. For example, if you are creating the entry corresponding to the suffix

`ou=people,dc=example,dc=com`, then you can choose the `organizationalUnit` object class or another object class that allows the `ou` attribute.

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

The **Property Editor** for the new entry opens. You can either accept the current values by clicking **OK** or modify the entry, as explained in [Section 1.3, “Modifying Directory Entries”](#).

1.2. Creating Directory Entries

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

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

[Table 2.1, “Entry Templates and Corresponding Object Classes”](#) shows what type of object class is used for each template.

Template	Object Class
User	<code>inetOrgPerson</code>
Group	<code>groupOfUniqueNames</code>
Organizational Unit	<code>organizationalUnit</code>

Template	Object Class
Role	nsRoleDefinition
Class of Service	cosSuperDefinition

Table 2.1. Entry Templates and Corresponding Object Classes

These templates contain fields representing all the mandatory attributes and some of the commonly used optional attributes. To create an entry using one of these templates, refer to [Section 1.2.1, “Creating an Entry Using a Predefined Template”](#). To create any other type of entry, refer to [Section 1.2.2, “Creating Other Types of Entries”](#).

1.2.1. Creating an Entry Using a Predefined Template

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

For information on starting the Directory Server Console, see [Section 4, “Starting the Directory Server Console”](#).

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

The corresponding **Create** window opens.

3. Supply values for all of the mandatory attributes (identified by an asterisk) and, if you want, for any of the optional attributes.
4. The **Create** window does not provide fields for all optional attributes. To display the full list of attributes, click the **Advanced** button.

In the **Property Editor** (described in [Section 1.3, “Modifying Directory Entries”](#)), select any additional attributes, and fill in the attribute values.

5. Click **OK** to save the entry. The new entry opens in the right pane.

1.2.2. Creating Other Types of Entries

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

For information on starting the Directory Server Console, see [Section 4, “Starting the Directory Server Console”](#).

2. In the left pane, right-click the main entry under which to add the new entry, and select **Other**.

The **New Object** window opens.

3. In the object class list, select an object class to define the new entry.

4. Click **OK**.

If you selected an object class related to a type of entry for which a predefined template is available, the corresponding **Create** window opens, as described in [Section 1.2.1, “Creating an Entry Using a Predefined Template”](#).

In all other cases, the **Property Editor** opens. It contains a list of mandatory attributes for the selected object class.

5. Supply a value for all the listed attributes.

- Some fields are empty, but some might have a placeholder value such as `New`. Fill in all attributes with a meaningful value for the entry.
- Some object classes can have several naming attributes. Select the naming attribute to use to name the new entry.
- Open the **Property Editor** to add optional attributes, as described in [Section 1.3, “Modifying Directory Entries”](#).

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

1.3. Modifying Directory Entries

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

- Add and remove object classes
- Add and remove an attribute
- Add and remove an attribute value
- Add an attribute subtype

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

1.3.1. Displaying the Property Editor

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

- From the **Directory** tab, by right-clicking an entry, and selecting **Properties** from the pop-up menu.

- From the **Directory** tab, by double-clicking an entry
- From the **Create...** entry templates, by clicking the **Advanced** button (as in [Section 1.2.1, “Creating an Entry Using a Predefined Template”](#))
- From the **New Object** window, by clicking **OK** (as in [Section 1.2.2, “Creating Other Types of Entries”](#))

1.3.2. Adding an Object Class to an Entry

To add an object class to an entry, do the following:

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

Alternatively, double-click the entry to open the **Property Editor**, and click the **Advanced** button.

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

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

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

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

4. Click **OK** in the **Advanced Property Editor** when you have finished editing the entry, then click **OK** to close the **Property Editor**.

1.3.3. Removing an Object Class

To remove an object class from an entry, do the following:

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

Alternatively, double-click the entry to open the **Property Editor** opens, and click the **Advanced** button.

2. Click the text box that shows the object class to remove, and then click **Delete Value**.
3. Click **OK** in the **Advanced Property Editor**, then click **OK** to save the changes and close the **Property Editor**.

1.3.4. Adding an Attribute to an Entry

Before you can add an attribute to an entry, the entry must contain an object class that either requires or allows the attribute. refer to [Section 1.3.2, “Adding an Object Class to an Entry”](#) and [Chapter 9, Extending the Directory Schema](#) for more information.

Add an attribute to an entry as follows:

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

Alternatively, double-click the entry to open the **Property Editor**, and then click the **Advanced** button.

2. Click **Add Attribute**. The Add Attribute dialog box opens.

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

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

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

5. Click **OK** in the **Advanced Property Editor** to save the attribute to the entry and close the **Advanced Property Editor**.

Click **OK** to close the **Property Editor**.



NOTE

If the attribute you want to add is not listed, add the object class containing the attribute first, then add the attribute. See [Section 1.3.2, “Adding an Object Class to an Entry”](#) for instructions on adding an object class.

1.3.5. Adding Very Large Attributes

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

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

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

- The size of each attribute name in the request
- The size of the values of each of the attributes in the request
- The size of the DN in the request
- Some overhead; usually 10 kilobytes is sufficient

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

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

1.3.6. Adding Attribute Values

Multi-valued attributes allow multiple value for one attribute to be added to an entry. To add an attribute value to a multi-valued attribute:

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

Alternatively, double-click the entry to open the **Property Editor**, and click the **Advanced** button.

2. Select the attribute to which to add a value, and then click **Add Value**. A new blank text field opens in the right column.
3. Type in the new attribute value.
4. Click **OK** in the **Advanced Property Editor** to close the **Advanced Property Editor**, then click **OK** again to close the **Property Editor**.

1.3.7. Removing an Attribute Value

To remove an attribute value from an entry, do the following:

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

Alternatively, double-click the entry to open the **Property Editor**, and click the **Advanced** button.

2. Click the text box of the attribute value to remove, and click **Delete Value**.

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

3. Click **OK** to close the **Advanced Property Editor**, then click **OK** to close the **Property Editor**.

1.3.8. Adding an Attribute Subtype

There are three different kinds of subtypes to attributes which can be added to an entry: language, binary, and pronunciation.

1.3.8.1. Language Subtype

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

```
givenname:lang-ja
```

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

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

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

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

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

Instead, use:

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

1.3.8.2. Binary Subtype

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

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

1.3.8.3. Pronunciation Subtype

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

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

1.3.8.4. Adding a Subtype to an Attribute

To add a subtype to an entry, do the following:

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

Alternatively, double-click the entry to open the **Property Editor**.

2. Click **Add Attribute**. The **Add Attribute** dialog box opens.
3. Select the attribute to add from the list.
4. To assign a language subtype to the attribute, select the subtype from the **Language** drop-down list.

Assign one of the other two subtypes, binary or pronunciation, from the **Subtype** drop-down list.

5. Click **OK** to close the **Add Attribute** window, then click **OK** again to close the **Property Editor**.

1.4. Deleting Directory Entries

To delete entries using the Directory Server Console, do the following:

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

For information on starting the Directory Server Console, see [Section 4, "Starting the Directory Server Console"](#).

2. Right-click the entry to delete in the navigation tree or in the right pane. To select multiple

entries, use **Ctrl** or **Shift**.

3. Select **Delete** from the **Edit** menu.



WARNING

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

2. Managing Entries from the Command-Line

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

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

- [Section 2.1, “Providing Input from the Command-Line”](#)
- [Section 2.2, “Creating a Root Entry from the Command-Line”](#)
- [Section 2.3, “Adding Entries Using LDIF”](#)
- [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#)
- [Section 2.5, “Deleting Entries Using `ldapdelete`”](#)
- [Section 2.6, “Using Special Characters”](#)



NOTE

You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for your directory, see [Chapter 6, Managing Access Control](#).

2.1. Providing Input from the Command-Line

When you provide input to the `ldapmodify` and `ldapdelete`¹ utilities directly from the command-line, you must use LDIF statements. For detailed information on LDIF statements, see [Section 4, “LDIF Update Statements”](#).

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

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

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

```
ldapmodify -D bindDN -w password -h hostname
dn: cn=Barry Nixon, ou=people, dc=example,dc=com
changetype: modify
delete: telephonenumber
-
add: manager
manager: cn=Harry Cruise, ou=people, dc=example,dc=com
^D
```

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

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

2.2. Creating a Root Entry from the Command-Line

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

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

The `ldapmodify` utility binds to the server and prepares it to add an entry. The new root object can then be added, as follows:

```
dn: Suffix_Name
objectclass: newobjectclass
```

¹ The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

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



NOTE

You can use `ldapmodify` to add root objects only if you have one database per suffix. If you create a suffix that is stored in several databases, you must use the `ldif2db` utility with the `-nooption` parameter to specify the database that will hold the new entries. For information, see [Section 1.3, “Importing from the Command-Line”](#).

2.3. Adding Entries Using LDIF

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

1. Define the entries in an LDIF file.

LDIF files are described in [Appendix A, LDAP Data Interchange Format](#).

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

See [Section 1.1, “Importing a Database from the Console”](#) for information about LDIF file formats. When you import the LDIF file, select **Append to database** in the **Import** dialog box so that the server will only import entries that do not currently exist in the directory.

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

2.4. Adding and Modifying Entries Using `ldapmodify`

The `ldapmodify`¹ command can add and modify entries in an existing Directory Server database. The `ldapmodify` command opens a connection to the specified server using the supplied distinguished name and password and modifies the entries based on LDIF update statements contained in a specified file. Because `ldapmodify` uses LDIF update statements, `ldapmodify` can do everything that `ldapdelete` can do.

Consider the following when using `ldapmodify`:

- If the server detects an attribute or object class in the entry that is not known to the server,

then the modify operation will fail when it reaches the erroneous entry. All entries that were processed before the error was encountered will be successfully added or modified. If you run `ldapmodify` with the `-c` option (do not stop on errors), all correct entries processed after the erroneous entry will be successfully added or modified.

- If a required attribute is not present, the modify operation fails. This happens even if the offending object class or attribute is not being modified.



NOTE

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

2.4.1. Adding Entries Using `ldapmodify`

Typically, to add the entries using `ldapmodify`, specify the DN and password to bind to the Directory Server, the port and host of the Directory Server, and the LDIF file to use. For example:

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

This `ldapmodify` example has the following values:

- The entries to be created are specified in the file `new.ldif`. (In this example, the LDIF statements in the `new.ldif` file do not specify a change type. They follow the format defined in [Section 1, “About the LDIF File Format”](#).)
- The Directory Manager is a database administrator who has the authority to modify the entries, and its password is `King-Pin`.
- The hostname is `cyclops`.
- The server uses port number 845.

[Table 2.2, “`ldapmodify` Parameters Used for Adding Entries”](#) describes the `ldapmodify` parameters used in the example.

Parameter Name	Description
<code>-a</code>	Specifies that the modify operation will add new entries to the directory.
<code>-D</code>	Specifies the distinguished name with which to authenticate to the server. The value must

Parameter Name	Description
	be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
-w	Specifies the password associated with the distinguished name specified in the <code>-D</code> parameter.
-h	Specifies the name of the host on which the server is running.
-p	Specifies the port number that the server uses.
-f	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from <code>stdin</code> . For information on supplying LDIF update statements from the command-line, refer to Section 2.1, “Providing Input from the Command-Line” .

Table 2.2. ldapmodify Parameters Used for Adding Entries

For full information on `ldapmodify` parameters, see the *Directory Server Configuration, Command, and File Reference*.

2.4.2. Modifying Entries Using ldapmodify

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

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

This `ldapmodify` example has the following values:

- The entries to modify are specified in the file `modify_statements`. Before the entries can be modified, you must first create the `modify_statements` file with the appropriate LDIF update statements; LDIF update statements are described in [Section 4, “LDIF Update Statements”](#).
- The bind DN is `cn=Directory Manager`, which has permissions to edit any entry in the database, and the password is `King-Pin`.

- The hostname is `cyclops`.
- The server uses port number 845.

Table 2.3, “*ldapmodify Parameters Used for Modifying Entries*” describes the `ldapmodify` parameters used in the example.

Parameter Name	Description
<code>-D</code>	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
<code>-w</code>	Specifies the password associated with the distinguished name specified in the <code>-D</code> parameter.
<code>-h</code>	Specifies the name of the host on which the server is running.
<code>-p</code>	Specifies the port number that the server uses.
<code>-f</code>	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from <code>stdin</code> . For information on supplying LDIF update statements from the command-line, refer to Section 2.1, “Providing Input from the Command-Line” .

Table 2.3. ldapmodify Parameters Used for Modifying Entries

For full information on `ldapmodify` parameters, see the *Directory Server Configuration, Command, and File Reference*.

2.5. Deleting Entries Using `ldapdelete`

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



NOTE

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

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

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

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

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

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

This `ldapdelete` example has the following values:

- The entries to delete have the DNs `cn=Robert Jenkins,ou=People,dc=example,dc=com` and `cn=Lisa Jangles,ou=People,dc=example,dc=com`.
- The bind DN is the Directory Manager, which has permission to delete every entry in the database, and a password of `King-Pin`.
- The hostname is `cyclops`.
- The server uses port number 845.

[Table 2.4, “*ldapdelete Parameters Used for Deleting Entries*”](#) describes the `ldapdelete` parameters used in the example:

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

Parameter Name	Description
-p	Specifies the port number that the server uses.

Table 2.4. ldapdelete Parameters Used for Deleting Entries

For full information on `ldapdelete` parameters, see the *Directory Server Configuration, Command, and File Reference*.

2.6. Using Special Characters

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

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

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

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

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

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

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

3. Tracking Modifications to Directory Entries

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

- *creatorsName*. The distinguished name of the person who initially created the entry.
- *createTimestamp*. The timestamp for when the entry was created in GMT (Greenwich Mean Time) format.
- *modifiersName*. The distinguished name of the person who last modified the entry.

- *modifyTimestamp*. The timestamp for when the entry was last modified in GMT format.



NOTE

When a database link is used by a client application to create or modify entries, the *creatorsName* and *modifiersName* attributes do not reflect the real creator or modifier of the entries. These attributes contain the name of the administrator who is granted proxy authorization rights on the remote server. For information on proxy authorization, see [Section 3.2.2.2, “Providing Bind Credentials”](#).

Do the following to enable the Directory Server to track when entries are created or modified:

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

The server adds the *creatorsName*, *createTimestamp*, *modifiersName*, and *modifyTimestamp* attributes to every newly created or modified entry.

4. Click **Save**.
5. Open the **Tasks** tab, and click **Restart Directory Server**.



NOTE

The Directory Server *must* be restarted for the changes to take effect.

4. LDIF Update Statements

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

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

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

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

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

The general format of LDIF update statements is as follows:

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

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

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

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

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

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

The following sections describe the change types in detail.

4.1. Adding an Entry Using LDIF

`changetype: add` adds an entry to the directory. When you add an entry, make sure to create an entry representing a branch point before you try to create new entries under that branch.

That is, to place an entry in a `People` and a `Groups` subtree, then create the branch point for those subtrees before creating entries within the subtrees. For example:

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

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

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

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

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

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

dn: ou=example.com Bolivia\, S.A.,dc=example,dc=com
```

```
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: example.com Bolivia\, S.A.

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

4.2. Renaming an Entry Using LDIF

`changetype: modrdn` changes an entry's relative distinguished name (RDN). An entry's RDN is the left-most element in the distinguished name. The RDN for `cn=Barry Nixon,ou=People,dc=example,dc=com` is `cn=Barry Nixon`, and the RDN for `ou=People,dc=example,dc=com` is `ou=People`. A `changetype: modrdn` operation changes that left-most value in an entry's DN.

The `modrdn` change type only changes the RDN; it cannot change other parts of a DN. For example, the entry `cn=Sue Jacobs,ou=People,dc=example,dc=com` can be changed to `cn=Susan Jacobs,ou=People,dc=example,dc=com`, but it cannot be modified to be `cn=Sue Jacobs,ou=old employees,dc=example,dc=com`.

The following command renames Sue Jacobs to Susan Jacobs:

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

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

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

4.2.1. A Note on Renaming Entries

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

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

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

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

4.3. Modifying an Entry Using LDIF

`changetype: modify` can add, replace, or remove attributes or attribute values in an entry. When you specify `changetype: modify`, you must also provide a change operation to indicate how the entry is to be modified. Change operations can be as follows:

- `add: attribute`

Adds the specified attribute or attribute value. If the attribute type does not currently exist for the entry, then the attribute and its corresponding value are created. If the attribute type already exists for the entry, then the specified attribute value is added to the existing value. If the particular attribute value already exists for the entry, then the operation fails, and the server returns an error.

- `replace: attribute`

The specified values are used to entirely replace the attribute's values. If the attribute does not already exist, it is created. If no replacement value is specified for the attribute, the attribute is deleted.

- `delete: attribute`

The specified attribute is deleted. If more than one value of an attribute exists for the entry, then all values of the attribute are deleted in the entry. To delete just one of many attribute values, specify the attribute and associated value on the line following the delete change operation.

This section contains the following topics:

- [Section 4.3.1, “Adding Attributes to Existing Entries Using LDIF”](#)
- [Section 4.3.2, “Changing an Attribute Value Using LDIF”](#)
- [Section 4.3.3, “Deleting All Values of an Attribute Using LDIF”](#)
- [Section 4.3.4, “Deleting a Specific Attribute Value Using LDIF”](#)

4.3.1. Adding Attributes to Existing Entries Using LDIF

Using `changetype: modify` with the `add` operation can add an attribute and an attribute value to an entry. For example, the following LDIF update statement adds a telephone number to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
```

The following example adds two telephone numbers to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
telephonenumber: 555-6789
```

The following example adds two `telephonenumber` attributes and a `manager` attribute to the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 555-1212
telephonenumber: 555-6789
-
add: manager
manager: cn=Sally Nixon,ou=People,dc=example,dc=com
```

The following example adds a `jpeg` photograph to the directory. The `jpeg` photo can be viewed in the Directory Server Gateway. In order to add this attribute to the directory, use the `-b` parameter, which indicates that `ldapmodify` should read the referenced file for binary values if the attribute value begins with a slash:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
```

```
changetype: modify
add: jpegphoto
jpegphoto: /path/to/photo
```

You can also add a jpeg photograph to the directory using the following standard LDIF notation:

```
jpegphoto: < file:/path/to/photo
```

Using the standard notation means that the `-b` parameter does not need to be used with `ldapmodify`. However, you must add `version:1` to the beginning of the LDIF file or with LDIF update statements. For example:

```
ldapmodify -D userDN -w user_password
version: 1
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate
userCertificate;binary:< file: BarneysCert
```



NOTE

Standard LDIF notation can *only* be used with the `ldapmodify` command, not with other command-line utilities.

4.3.2. Changing an Attribute Value Using LDIF

`changetype: modify` with the `replace` operation changes all values of an attribute in an entry. For example, the following LDIF update statement changes Barney's manager from Sally Nixon to Wally Hensford:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
replace: manager
manager: cn=Wally Hensford, ou=People, dc=example,dc=com
```

If the entry has multiple instances of the attribute, then to change one of the attribute values, you must delete the attribute value first and then add the replacement value. For example, this entry has two telephone numbers:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-1212
```

```
telephonenumber: 555-6789
```

To change the telephone number 555-1212 to 555-4321, use the following LDIF update statement:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
telephonenumber: 555-1212
-
add: telephonenumber
telephonenumber: 555-4321
```

The entry is now as follows:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-6789
telephonenumber: 555-4321
```

4.3.3. Deleting All Values of an Attribute Using LDIF

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

For example, the following LDIF update statement deletes all instances of the `telephonenumber` attribute from the entry, regardless of how many times it appears in the entry:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
```

To delete just a specific instance of the `telephonenumber` attribute, simply delete that specific attribute value, as described in the next section.

4.3.4. Deleting a Specific Attribute Value Using LDIF

Running `changetype: modify` with the `delete` operation can delete a single value for an attribute value from an entry, as well as deleting all instances of the attribute. For example, consider the following entry:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
```

```
telephonenumber: 555-1212
telephonenumber: 555-6789
```

To delete the 555-1212 telephone number from this entry, use the following LDIF update statement:

```
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
delete: telephonenumber
telephonenumber: 555-1212
```

Barney's entry then becomes:

```
cn=Barney Fife,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-6789
```

4.4. Deleting an Entry Using LDIF

`changetype: delete` is the change type which deletes an entire entry from the directory.



NOTE

You can only delete leaf entries. Therefore, when you delete an entry, make sure that no other entries exist under that entry in the directory tree. That is, you cannot delete an organizational unit entry unless you have first deleted all the entries that belong to the organizational unit.

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

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

The entry that identifies the `People` subtree can be deleted only if no other entries exist below it.

The following LDIF update statements can be used to delete person entries:

```
dn: cn=Pete Minsky,ou=People,dc=example,dc=com
changetype: delete
dn: cn=Sue Jacobs,ou=People,dc=example,dc=com
changetype: delete
```

**CAUTION**

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

4.5. Modifying an Entry in an Internationalized Directory

If the attribute values in the directory are associated with languages other than English, the attribute values are associated with language tags. When using the `ldapmodify` command-line utility to modify an attribute that has an associated language tag, you must match the value and language tag exactly or the modify operation will fail.

For example, to modify an attribute value that has a language tag of `lang-fr`, include `lang-fr` in the modify operation, as follows:

```
dn: bjensen,dc=example,dc=com
changetype: modify
replace: homePostalAddress;lang-fr
homePostalAddress;lang-fr: 34 rue de Seine
```

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 may refer to the updated entry.

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

5.1. How Referential Integrity Works

When the Referential Integrity Plug-in (see [Section 1.26, "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 Plug-in is disabled.

**NOTE**

The Referential Integrity Plug-in should only be enabled on one supplier replica

in a multi-master replication environment to avoid conflict resolution loops. When enabling the plug-in on servers issuing chaining requests, be sure to analyze performance resource and time needs, as well as your integrity needs. Integrity checks can be time-consuming and draining on memory and CPU.

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

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

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

All attributes used in referential integrity *must* be indexed for presence and equality; not indexing those attributes results poor server performance for modify and delete operations. See [Section 2, “Creating 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 Plug-in in a replication environment:

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

If the replication environment satisfies all of those conditions, you can enable the Referential Integrity Plug-in.

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



NOTE

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

5.3. Enabling/Disabling Referential Integrity

You can enable or disable referential integrity as follows:

1. Start the Directory Server Console. See [Section 4, “Starting the Directory Server Console”](#).
2. Select the **Configuration** tab.
3. Expand the **Plugins** folder in the navigation tree, and select **Referential Integrity Postoperation Plug-in** from the list.

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

4. Check the **Enable plugin** checkbox to enable the plug-in; clear it to disable it.
5. Click **Save**.
6. For your changes to be applied, go to the **Tasks** tab, and select **Restart the Directory Server**.

5.4. Modifying the Update Interval

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

- Update immediately
- 90 seconds
- 3600 seconds (updates occur every hour)
- 10,800 seconds (updates occur every 3 hours)
- 28,800 seconds (updates occur every 8 hours)
- 86,400 seconds (updates occur once a day)
- 604,800 seconds (updates occur once a week)

To modify the update interval, do the following:

1. Start the Directory Server Console. See [Section 4, “Starting the Directory Server Console”](#).
2. Select the **Configuration** tab.
3. Expand the **Plugins** folder in the navigation tree, and select the **Referential Integrity Postoperation Plug-in**.

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

4. In the arguments list, replace the value in the first text box with the appropriate time interval.
5. Click **Save**.
6. For your changes to be applied, go to the **Tasks** tab, and click **Restart the Directory Server**.

5.5. 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 through the Directory Server Console, such as adding the *nsroledn* attribute if roles are being used.



NOTE

Keep in mind that any attribute specified in the Referential Integrity Plug-in parameter list *must* have equality indexing on all databases. Otherwise, the plug-in scans every entry of the databases for matching the deleted or modified DN, degrading performance severely. If you add an attribute, ensure that it is indexed in all the backends.



TIP

Improve the performance by removing any unused attributes from the list.

1. Start the Directory Server Console. See [Section 4, “Starting the Directory Server Console”](#).
2. Select the **Configuration** tab.
3. Expand the **Plugins** folder in the navigation tree, and select the **Referential Integrity Postoperation Plug-in**.

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

4. In the **Arguments** section, use the **Add** and **Delete** buttons to modify the attributes in the list.
5. Click **Save**.
6. For your changes to be applied, go to the **Tasks** tab, and select **Restart the Directory Server**.



NOTE

All attributes used in referential integrity *must* be indexed for presence and equality; not indexing those attributes results poor server performance for modify and delete operations. See [Section 2, “Creating Indexes”](#) for more information about checking and creating indexes.

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.

1. Creating and Maintaining Suffixes

Different pieces of the directory tree can be stored in different databases, and then these databases can be distributed across multiple servers. The directory tree contains branch points called *nodes*. These nodes may be associated with databases. A suffix is a node of the directory tree associated with a particular database. For example, a simple directory tree might appear as illustrated in [Figure 3.1](#), “A Sample Directory Tree with One Root Suffix”.

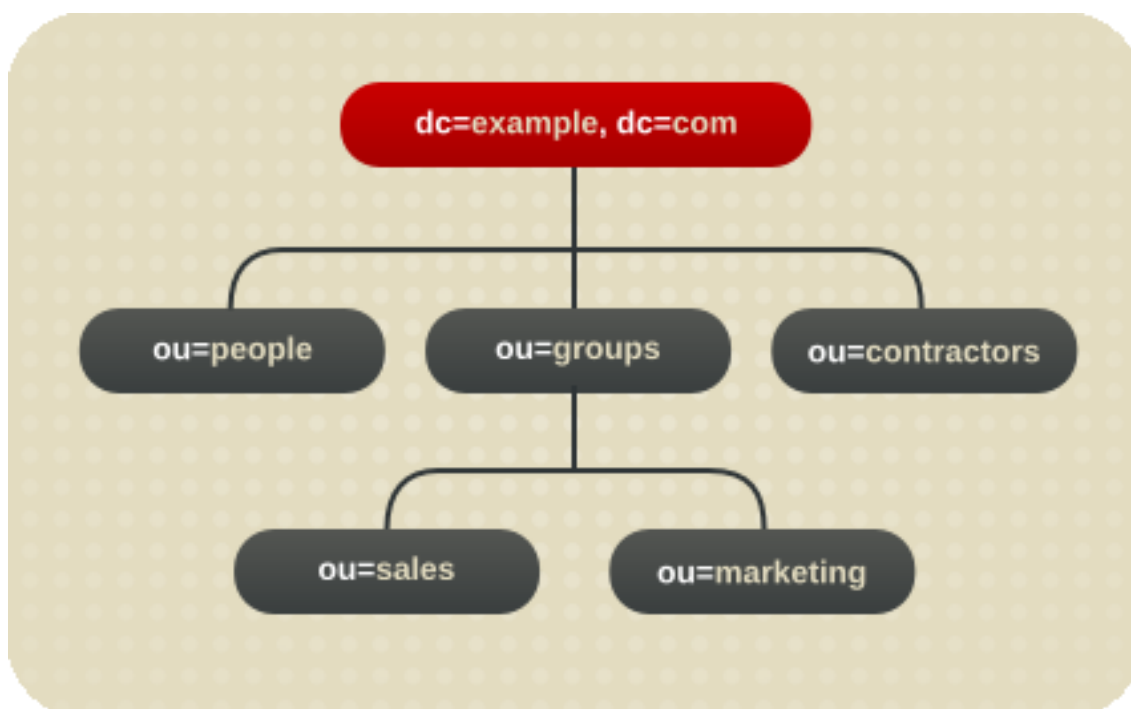


Figure 3.1. A Sample Directory Tree with One Root Suffix

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

This section describes creating suffixes on Directory Server and associating them with databases.

- [Section 1.1, “Creating Suffixes”](#)
- [Section 1.2.1, “Using Referrals in a Suffix”](#)

1.1. Creating Suffixes

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

A directory might contain more than one root suffix. For example, an ISP might host several websites, one for `example.com` and one for `redhat.com`. The ISP would create two root suffixes, one corresponding to the `dc=example,dc=com` naming context and one corresponding to the `dc=redhat,dc=com` naming context, as shown in [Figure 3.2, “A Sample Directory Tree with Two Root Suffixes”](#).

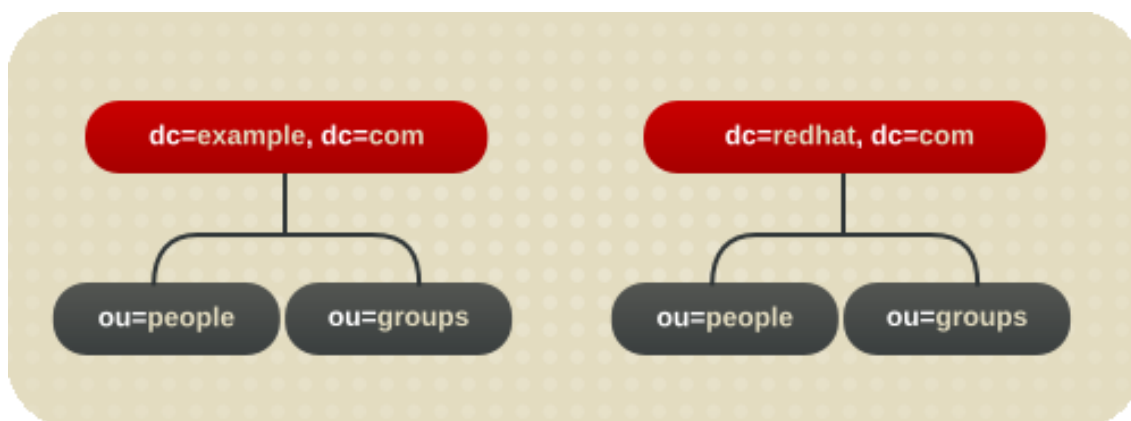


Figure 3.2. A Sample Directory Tree with Two Root Suffixes

It is also possible to create root suffixes to exclude portions of the directory tree from search operations. For example, Example Corporation wants to exclude their European office from a search on the general Example Corporation directory. To do this, they create two root suffixes. One root suffix corresponds to the general Example Corporation directory tree, `dc=example,dc=com`, and one root suffix corresponds to the European branch of their directory tree, `l=europe,dc=example,dc=com`. From a client application's perspective, the directory tree looks as illustrated in [Figure 3.3, “A Sample Directory Tree with a Root Suffix Off Limits to Search Operations”](#).

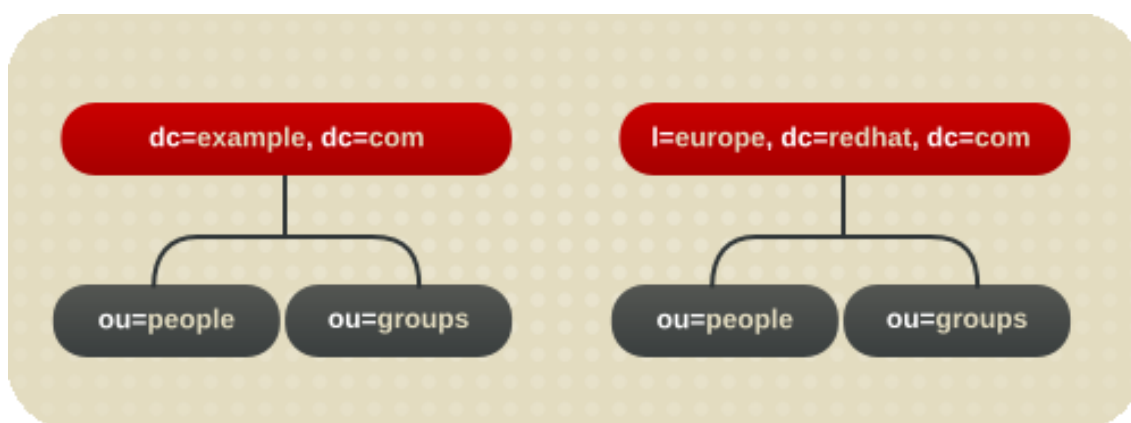


Figure 3.3. A Sample Directory Tree with a Root Suffix Off Limits to Search Operations

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

If Example Corporation decides to include the entries in the European branch of their directory tree in general searches, they make the European branch a sub suffix of the general branch. To do this, they create a root suffix for Example Corporation, `dc=example, dc=com`, and then create a sub suffix beneath it for their European directory entries, `l=europe, dc=example, dc=com`. From a client application's perspective, the directory tree appears as illustrated in [Figure 3.4, "A Sample Directory Tree with a Sub Suffix"](#).

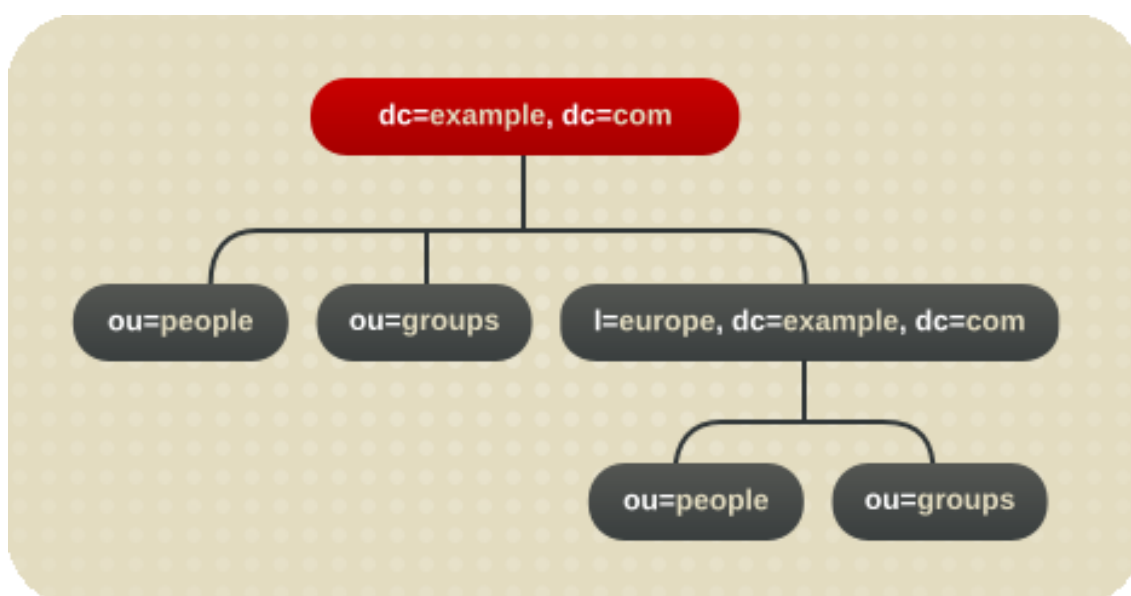


Figure 3.4. A Sample Directory Tree with a Sub Suffix

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

- [Section 1.1.1, “Creating a New Root Suffix Using the Console”](#)
- [Section 1.1.2, “Creating a New Sub Suffix Using the Console”](#)
- [Section 1.1.3, “Creating Root and Sub Suffixes from the Command-Line”](#)

1.1.1. Creating a New Root Suffix Using the Console

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

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

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

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

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

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

5. Click **OK**.

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

1.1.2. Creating a New Sub Suffix Using the Console

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

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

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

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

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

Deselect the checkbox to create a database for the new sub suffix later. The new sub suffix will be disabled until a database is created.

5. Click **OK**.

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

1.1.3. Creating Root and Sub Suffixes from the Command-Line

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



NOTE

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

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

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


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

2. Create the root suffix entry. For example:

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

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

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



NOTE

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

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

Attribute Name	Value
dn	Defines the DN for the suffix. The DN is contained in quotes. The value entered takes the form <code>cn="dc=domain,dc=com",cn=mapping tree, cn=config</code> . This attribute is required.
cn	Defines the relative DN (RDN) of the entry. This attribute is required.
objectclass	Tells the server that the entry is root or sub suffix entry. It always takes the value <code>nsMappingTree</code> . This attribute is required.
nsslapd-state	Determines how the suffix handles operations. This attribute takes the following values: <ul style="list-style-type: none">• <code>backend</code>: The backend (database) is used to process all operations.

Attribute Name	Value
	<ul style="list-style-type: none"> <code>disabled</code>: The database is not available for processing operations. The server returns a <i>No such search object</i> error in response to requests made by client applications. <code>referral</code>: A referral is returned for requests made to this suffix. <code>referral on update</code>: The database is used for all operations except update requests, which receive a referral. <p>The default value is <code>disabled</code>.</p>
<code>nsslapd-referral</code>	Defines the LDAP URL of the referral to be returned by the suffix. This attribute can be multi-valued, with one referral per value. This attribute is required when the value of the <code>nsslapd-state</code> attribute is <code>referral</code> or <code>referral on update</code> .
<code>nsslapd-backend</code>	Gives the name of the database or database link used to process requests. This attribute can be multi-valued, with one database or database link per value. See Section 3, “Creating and Maintaining Database Links” for more information about database links. This attribute is required when the value of the <code>nsslapd-state</code> attribute is set to <code>backend</code> or <code>referral on update</code> .
<code>nsslapd-distribution-plugin</code>	Specifies the shared library to be used with the custom distribution function. This attribute is required only when more than one database is specified in the <code>nsslapd-backend</code> attribute. See Section 2, “Creating and Maintaining Databases” for more information about the custom distribution function.
<code>nsslapd-distribution-funct</code>	Specifies the name of the custom distribution function. This attribute is required only when more than one database is specified in the <code>nsslapd-backend</code> attribute. See Section 2, “Creating and Maintaining Databases” for more information about the custom distribution function.
<code>nsslapd-parent-suffix</code>	Provides the DN of the parent entry for a sub suffix. By default, this attribute is not present,

Attribute Name	Value
	which means that the suffix is regarded as a root suffix. For example, to create a sub suffix names <code>o=sales,dc=example,dc=com</code> under the root suffix <code>dc=example,dc=com</code> , add <code>nsslapd-parent-suffix:</code> <code>"dc=example,dc=com"</code> to the sub suffix.

Table 3.1. Suffix Attributes

1.2. Maintaining Suffixes

This section describes the following procedures:

- [Section 1.2.1, “Using Referrals in a Suffix”](#)
- [Section 1.2.2, “Enabling Referrals Only During Update Operations”](#)
- [Section 1.2.3, “Disabling a Suffix”](#)
- [Section 1.2.4, “Deleting a Suffix”](#)

1.2.1. Using Referrals in a Suffix

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

To set referrals in a suffix, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left pane, select the suffix for which to add a referral.
3. Click the **Suffix Settings** tab, and select the **Return Referrals for all Operations** radio button.
4. Click the **Referrals** tab. Enter an LDAP URL in the **Enter a new referral** field, or click **Construct** to be guided through the creation of an LDAP URL. [Appendix C, LDAP URLs](#) has more information about the structure of LDAP URLs.
5. Click **Add** to add the referral to the list.

You can enter multiple referrals. The directory will return the entire list of referrals in response

to requests from client applications.

6. Click **Save**.

1.2.2. Enabling Referrals Only During Update Operations

It is possible to configure the directory to redirect update and write requests made by client applications to a read-only database. For example, there may be a local copy of directory data, and that data should be available company-wide for searches but not for updates. Enabling referrals for that Directory Server only for update requests means that when a client application asks to update an entry, the client is referred to the server that owns the data, where the modification request can proceed.

To enable referrals only during update operations, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left pane, click the suffix to which to add a referral.
3. Click the **Suffix Settings** tab, and select the **Return Referrals for Update Operations** radio button.
4. Click the **Referrals** tab. Enter an LDAP URL in the **Enter a new referral** field, or click **Construct** to be guided through the creation of an LDAP URL.

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

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

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

6. Click **Save**.

1.2.3. Disabling a Suffix

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

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

To disable a suffix, do the following:

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

2. Under **Data** in the left navigation pane, click the suffix to disable.
3. Click the **Suffix Setting** tab, and deselect the **Enable this suffix** checkbox.
4. Click **Save**.

The suffix is no longer enabled.

1.2.4. Deleting a Suffix



CAUTION

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

1. In the Directory Server Console, select the **Configuration** tab.
2. Under **Data** in the left navigation pane, select the suffix to delete.
3. Select **Delete** from the **Object** menu.

Alternatively, right-click the suffix and select **Delete** from the pop-up menu.
4. Select **Delete this suffix and all of its sub suffixes** to remove all the suffix and every suffix below it.

Select **Delete this suffix only** to remove only this particular suffix, not its sub suffixes.
5. Click **OK**.

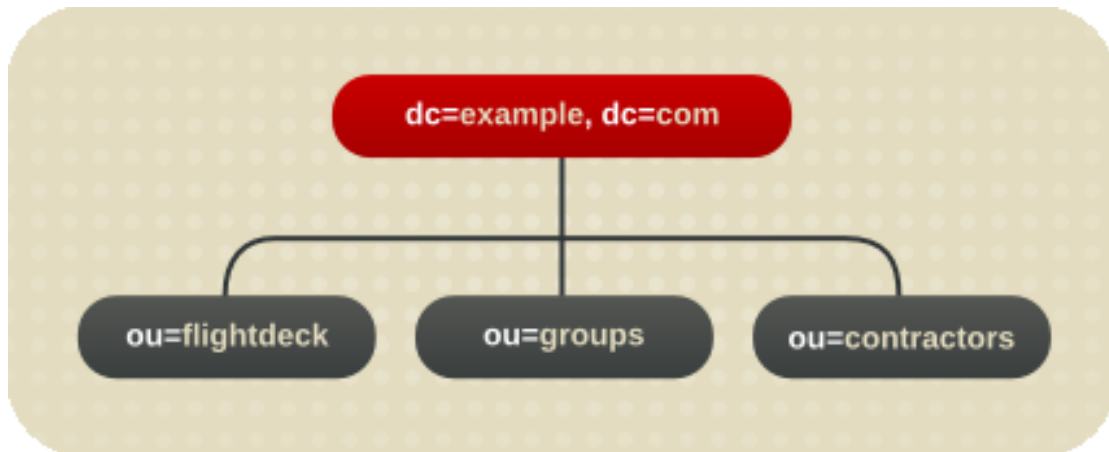
2. Creating and Maintaining Databases

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

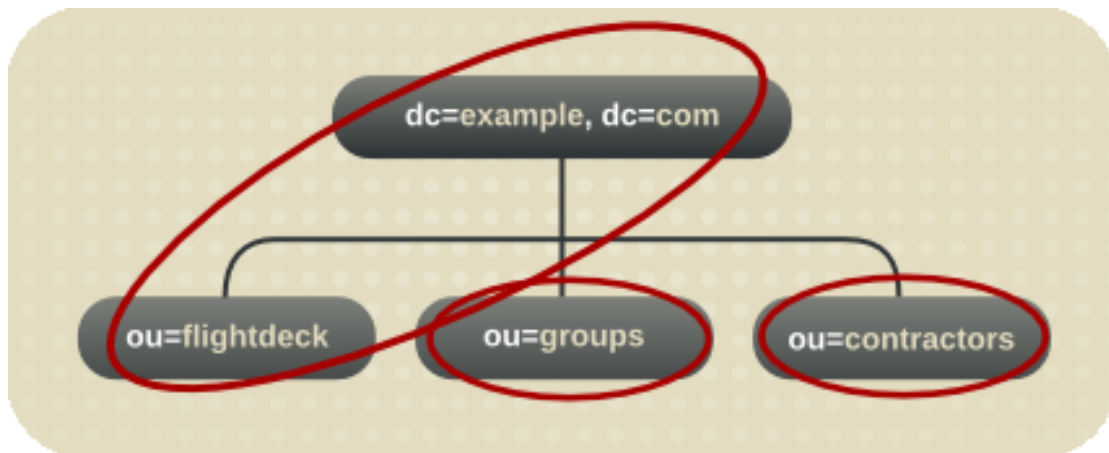
2.1. Creating Databases

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

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



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



This division of the tree corresponds to three databases.

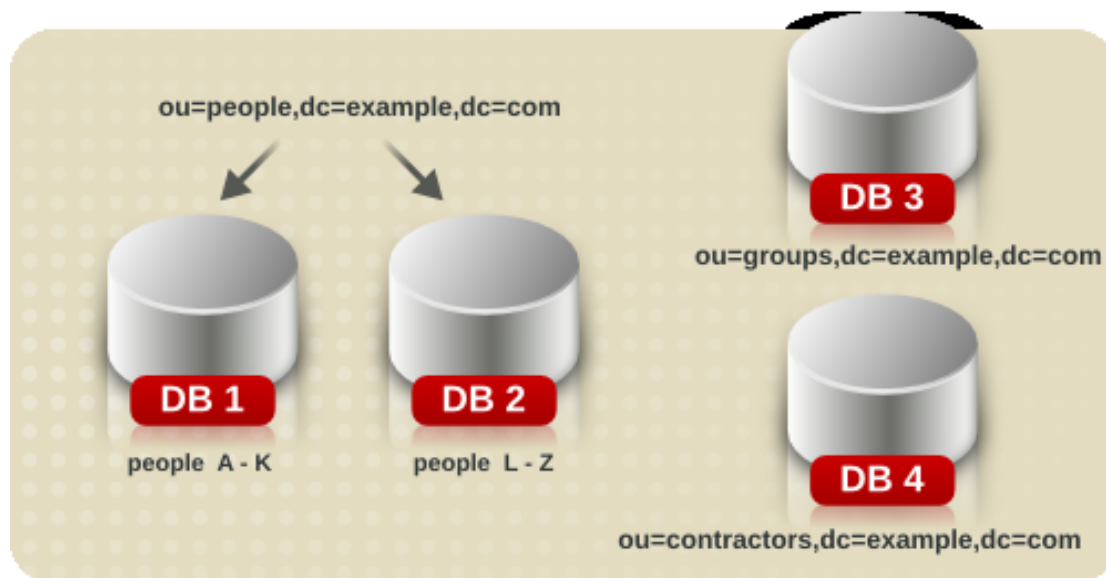


Database one contains the data for `ou=people` plus the data for `dc=example,dc=com`, so that

clients can conduct searches based at `dc=example,dc=com`. Database two contains the data for `ou=groups`, and database three contains the data for `ou=contractors`.

- Multiple databases for one suffix.

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



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

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

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

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

The **Create New Database** dialog box is displayed.

4. In the **Create New Database** dialog box, enter a unique name for the database, such as `example2`. The database name can be a combination of alphanumeric characters, dashes (-), and underscores (_). No other characters are allowed.

5. In the **Create database in** field, enter the path to the directory to store the new database. Alternatively, click **Browse** to locate a directory on the local machine.

By default, the directory stores the new database in the
`/var/lib/dirsrv/slapd-instance_name/db` directory.

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

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

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

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

1. Run `ldapmodify`.¹

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

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

2. Create the entry for the new database.

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

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

3. Create a root or sub suffix, as described in [Section 1.1.3, “Creating Root and Sub Suffixes from the Command-Line”](#). The database name, given in the DN attribute, must correspond with the value in the `nsslapd-backend` attribute of the suffix entry.

2.1.3. Adding Multiple Databases for a Single Suffix

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



NOTE

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

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

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

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

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

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

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

The **Database List** dialog box is displayed. Select a database from the list, and click **OK**.

5. Enter the path to the distribution library in the **Distribution library** field, or click **Browse** to locate a distribution library on the local machine.
6. Enter the name of the distribution function in the **Function name** field.
7. Click **Save**.

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

1. Run `ldapmodify`.¹

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

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

```
nsslapd-backend: Database1
nsslapd-backend: Database2
nsslapd-backend: Database3
nsslapd-distribution-plugin: /full/name/of/a/shared/library
nsslapd-distribution-funct:distribution-function-name
```

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

For more information about using the `ldapmodify` command-line utility, see [Section 2.4](#), “*Adding and Modifying Entries Using ldapmodify*”.

2.2. Maintaining Directory Databases

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

- [Section 2.2.1](#), “*Placing a Database in Read-Only Mode*”
- [Section 2.2.2](#), “*Deleting a Database*”
- [Section 2.2.3](#), “*Configuring Transaction Logs for Frequent Database Updates*”

2.2.1. Placing a Database in Read-Only Mode

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

The Directory Server Console and the command-line utilities do not automatically put the directory in read-only mode before export or backup operations because this would make your

directory unavailable for updates. However, with multi-master replication, this might not be a problem.

- [Section 2.2.1.1, “Making a Database Read-Only Using the Console”](#)
- [Section 2.2.1.2, “Making a Database Read-Only from the Command Line”](#)
- [Section 2.2.1.3, “Placing the Entire Directory Server in Read-Only Mode”](#)

2.2.1.1. Making a Database Read-Only Using the Console

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

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

The change takes effect immediately.

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

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

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

To manually place a database into read-only mode, do the following:

1. Run `ldapmodify`.¹

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

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

```
dn: cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
```

```
replace: nsslapd-readonly  
nsslapd-readonly: on
```



NOTE

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

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

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



WARNING

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



NOTE

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

To put the Directory Server in read-only mode, do the following:

1. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
2. Select the **Settings** tab in the right pane.
3. Select the **Make Entire Server Read-Only** checkbox.
4. Click **Save**, and then restart the server.

2.2.2. Deleting a Database

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

the physical database itself.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation pane, locate the database to delete, and select it.
3. From the **Object** menu, select **Delete**.

Alternatively, right-click the database and select **Delete** from the pop-up menu.

The **Deleting Database** confirmation dialog box is displayed.

4. Click **Yes** to confirm the deletion.

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

2.2.3. Configuring Transaction Logs for Frequent Database Updates

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

1. Open the configuration directory.

```
cd /etc/dirsrv/slapd-instance_name
```

2. Edit the `dse.ldif` file, and change the `nsslapd-db-logdirectory` directive for the new log file path:

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

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

2.3. Database Encryption

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

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

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



NOTE

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

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

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

2.3.1. Encryption Keys

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

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



CAUTION

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



CAUTION

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

2.3.2. Encryption Ciphers

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

The following ciphers are supported:

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

All ciphers are used in Cipher Block Chaining mode.

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

2.3.3. Configuring Database Encryption from the Console

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

encrypt.



NOTE

For existing attribute entries to be encrypted, the information must be exported, then re-imported. See [Section 2.3.5, “Exporting and Importing an Encrypted Database”](#).

7. Select which encryption cipher to use.
8. Repeat steps 6 and 7 for every attribute to encrypt. Then hit **Save**.

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

2.3.4. Configuring Database Encryption Using the Command-Line

1. Run the `ldapmodify` command¹:

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

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

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

3. For existing attributes in entries to be encrypted, the information must be exported, then re-imported. See [Section 2.3.5, “Exporting and Importing an Encrypted Database”](#).

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

2.3.5. Exporting and Importing an Encrypted Database

Exporting and importing encrypted databases is similar to exporting and importing regular databases. However, the encrypted information must be decrypted when it is exported to LDIF,

then re-encrypted when it is imported to the database. Using the `-E` option when running the `db2ldif` and `ldif2db` scripts will decrypt the data on export and re-encrypt it on import.

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

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

See [Section 2.3, “Exporting to LDIF from the Command-Line”](#) for more information.

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

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

See [Section 1.3, “Importing from the Command-Line”](#) for more information.



NOTE

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

- It is possible for old, unencrypted data to persist in the server's database page pool backing file, even after a successful re-import with encryption. To remove this data, stop the server and delete the `db/guardian` file, then re-start the server. This will force recovery, a side-effect of which is deleting the backing file. However, it is possible that the data from the deleted file could still be recovered from the hard drive unless steps are taken to overwrite the disk blocks that it occupied.
- After enabling encryption and importing data, be sure to delete the LDIF file because it contains plain text values for the now-encrypted data. Ensure that the disk blocks that it occupied are overwritten.
- The unencrypted data previously stored in the server's database may persist on disk after a successful re-import with encryption. This is because the old database files are deleted as part of the import process. Ensure that the disk blocks that those files occupied are overwritten.
- Data stored in the server's replication log database is never encrypted; therefore, care should be taken to protect those files if replication is used.

- The server does not attempt to protect unencrypted data stored in memory. This data may be copied into a system page file by the operating system. For this reason, ensure that any page or swap files are adequately protected.

3. Creating and Maintaining Database Links

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

[Section 5, “Monitoring Database Link Activity”](#) covers how to monitor database link activity.

- [Section 3.1, “Configuring the Chaining Policy”](#)
- [Section 3.2, “Creating a New Database Link”](#)
- [Section 3.3, “Chaining Using SSL”](#)
- [Section 3.4, “Maintaining Database Links”](#)
- [Section 3.5, “Database Links and Access Control Evaluation”](#)
- [Section 3.6, “Advanced Feature: Tuning Database Link Performance”](#)
- [Section 3.7, “Advanced Feature: Configuring Cascading Chaining”](#)

3.1. 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.

3.1.1. Chaining Component Operations

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

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

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

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

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

Component Name	Description	Permissions
ACI plug-in	This plug-in implements access control. Operations used to retrieve and update <i>ACI</i> 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, <i>nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config.</i>	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, <i>nsActiveChainingComponents: cn=resource limits,cn=components,cn=config.</i>	Read, search, and compare
Certificate-based authentication checking component	This component is used when the SASL-external bind method is used. It retrieves the user certificate from the database on the remote server. Allowing this component to chain means certificate-based	Read, search, and compare

Component Name	Description	Permissions
	authentication can work with a database link. To chain this component's operations, add the chaining component attribute, <i>nsActiveChainingComponents: cn=certificate-based authentication,cn=components,</i>	<i>cn=config.</i>
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, <i>nsActiveChainingComponents: cn=referential integrity postoperation,cn=plugins,cn=config.</i>	Read, write, search, and compare
Attribute Uniqueness plug-in	This plug-in checks that all the values for a specified <i>uid</i> attribute are unique (no duplicates). If this plug-in is chained, it confirms that the <i>uid</i> attribute values are unique even on attributes changed through a database link. To chain this component's operations, add the chaining component attribute, <i>nsActiveChainingComponents: cn=attribute uniqueness,cn=plugins,cn=config</i>	Read, search, and compare

Table 3.2. Components Allowed to Chain



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 1.4, “ACI Limitations”](#).

After modifying the components allowed to chain, restart the server in order for the modification to take effect.

3.1.1.1. Chaining Component Operations Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand **Data** in the left pane, and click **Database Link Settings**.
3. Select the **Settings** tab in the right window. To add a component to the **Components allowed to chain** list, click **Add**.

The **Select Components to Add** dialog box displays. Select a component from the list, and click **OK**.

4. To delete a component from the list, select it, and click **Delete**.
5. Click **Save**.
6. Restart the server in order for the change to take effect.

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

plug-in:

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

3.1.1.2. Chaining Component Operations from the Command-Line

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

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

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

See [Table 3.2, “Components Allowed to Chain”](#) for a list of the components which can be chained.

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

```
service dirsrv restart instance
```

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

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

3.1.2. 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.

² To ensure side effecting, this control sorts entries according to their attribute values. See [Section 3, “Starting and Stopping Servers”](#).

- *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 3.7.5, “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.

3.1.2.1. Chaining LDAP Controls Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder in the left pane, and click **Database Link Settings**.
3. Select the **Settings** tab in the right window. To add an LDAP control to the list, click **Add**.

The **Select control OIDs to add** dialog box displays. Select the OID of a control to add to the list, and click **OK**.

4. To delete a control from the list, select it from the **LDAP controls forwarded to the remote server** list, and click **Delete**.
5. Click **Save**.

3.1.2.2. Chaining LDAP Controls from the Command-Line

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

```
nsTransmittedControls: 2.16.840.1.113730.3.4.9
```

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

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

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

Table 3.3. LDAP Controls and Their OIDs

For more information about LDAP controls, refer to the LDAP C-SDK documentation at <http://www.mozilla.org/directory>.

3.2. Creating a New Database Link

The basic configuration of the database link involves 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.
- *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.

3.2.1. Creating a New Database Link Using the Console

To create a new database link using the Directory Server Console:

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

A **Create New Suffix** dialog box is displayed.

3. Enter the name of the suffix on the remote server to which to chain the suffix in the **New suffix** field.

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

4. Deselect the **Create associated database automatically** checkbox.

The checkbox must not be selected because a database link cannot be added to a suffix that is associated with a database. This suffix is used only by the database link.

5. Click **OK**.

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

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

The **Create New Database Link** dialog box is displayed.

7. Enter the name of the new database link in the **Database link name** field, such as `examplelink1`. The name can be a combination of alphanumeric characters, dashes (-), and underscores (_). No other characters are allowed.

8. Enter the DN used by the database link to bind to the remote server in the **Bind DN** field, such as `cn=dblink`.

9. Enter the password used by the database link to bind to the remote server in the **Password** field.

10. Select the **Use a secure LDAP connection between servers** checkbox for the database link to use SSL to communicate to the remote server.

11. Enter the name of the remote server in the **Remote server** field. Enter the server port number used for the bind in the **Remote server port** field. The default port number is 389. The default SSL port number is 636.

12. Enter the name of a failover server in the **Failover Server(s)** field, and specify a port number in the **Port** field. The default port number is 389. The default SSL port number is 636. Click **Add** to add the failover server to the list.

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

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

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



TIP

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

appears in the **Remote server checklist** box.

3.2.2. Creating a Database Link from the Command-Line

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

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

2. Specify the configuration information for the database link:

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

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

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

- [Section 3.2.2.1, “Providing Suffix Information”](#)
- [Section 3.2.2.2, “Providing Bind Credentials”](#)
- [Section 3.2.2.3, “Providing an LDAP URL”](#)
- [Section 3.2.2.4, “Providing a List of Failover Servers”](#)
- [Section 3.7.6, “Summary of Cascading Chaining Configuration Attributes”](#)
- [Section 3.2.2.6, “Database Link Configuration Example”](#)

3.2.2.1. Providing Suffix Information

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

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

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



NOTE

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

3.2.2.2. Providing Bind Credentials

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

Providing bind credentials involves the following steps:

1. On the remote server, do the following:

- Create an administrative user for the database link.

For information on adding entries, see [Chapter 2, Creating Directory Entries](#).

- Provide proxy access rights for the administrative user created in step 1 on the subtree chained to by the database link.

For more information on configuring ACIs, see [Chapter 6, Managing Access Control](#)

2. On the server containing the database link, use `ldapmodify` to provide a user DN for the database link in the `nsMultiplexorBindDN` attribute of the `cn=database_link`, `cn=chaining database`, `cn=plugins`, `cn=config` entry.

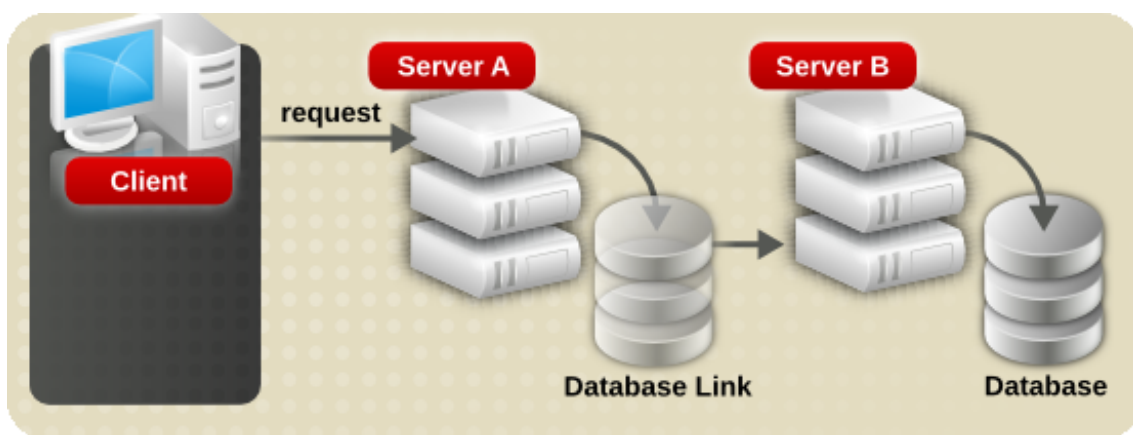


CAUTION

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

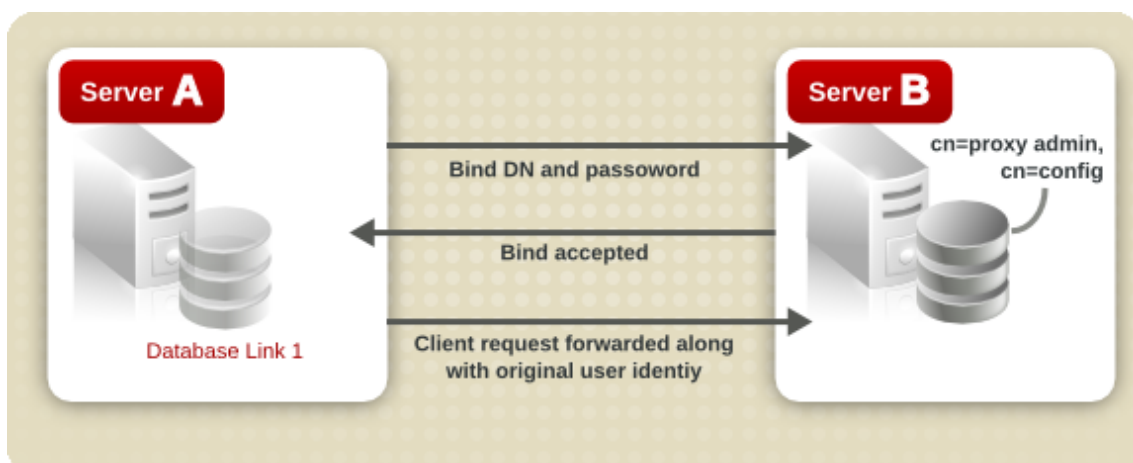
Use `ldapmodify` to provide a user password for the database link in the `nsMultiplexorCredentials` attribute of the `cn=database_link, cn=chaining database, cn=plugins, cn=config` entry.

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



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

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



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



CAUTION

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 via 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 6, Managing Access Control](#). For more information about the proxy authentication control, refer to the LDAP C-SDK documentation at <http://www.mozilla.org/directory>.



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.

3.2.2.3. Providing an LDAP URL

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

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

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

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

For the database link to connect to the remote server using LDAP over SSL, the LDAP URL of the remote server uses the protocol LDAPS instead of LDAP in the URL, such as `ldaps://example.com:636`.

For more information about chaining and SSL, see [Section 3.3, “Chaining Using SSL”](#).

3.2.2.4. Providing a List of Failover Servers

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

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

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

3.2.2.5. Summary of Database Link Configuration Attributes

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

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

Attributes	Value
<code>nsTransmittedControls</code> [†]	Gives the OID of LDAP controls forwarded by the database link to the remote data server.
<code>nsslapd-suffix</code>	The suffix managed by the database link. Any changes to this attribute after the entry has been created take effect only after the server containing the database link is restarted.
<code>nsslapd-timelimit</code>	Default search time limit for the database link, given in seconds. The default value is 3600 seconds.
<code>nsslapd-sizelimit</code>	Default size limit for the database link, given in number of entries. The default value is 2000 entries.
<code>nsFarmServerURL</code>	Gives the LDAP URL of the remote server (or farm server) that contains the data. This attribute can contain optional servers for failover, separated by spaces. If using cascading chaining, this URL can point to another database link.
<code>nsMultiplexorBindDN</code>	DN of the administrative entry used to communicate with the remote server. The term <i>multiplexor</i> in the name of the attribute means the server which contains the database link and communicates with the

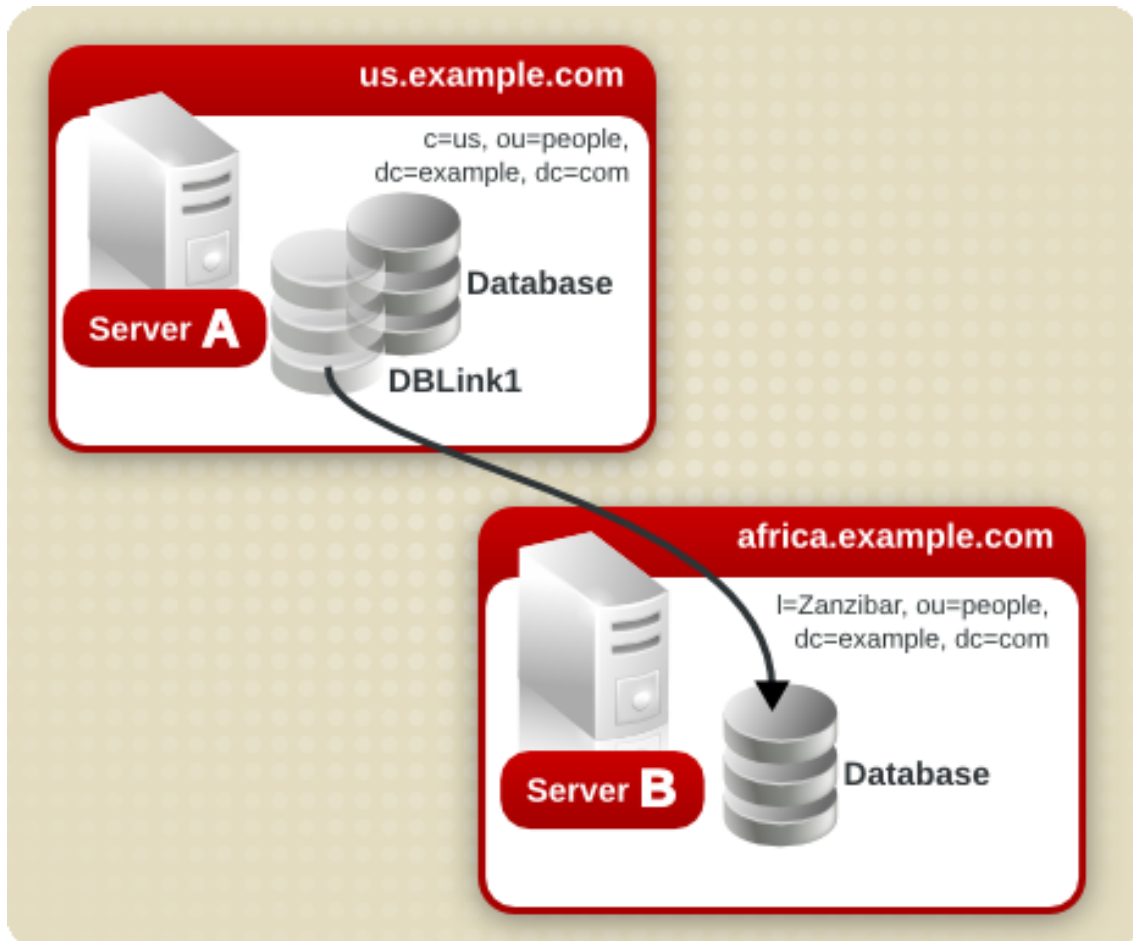
Attributes	Value
	remote server. This bind DN cannot be the Directory Manager. If this attribute is not specified, the database link binds as <code>anonymous</code> .
<code>nsMultiplexorCredentials</code>	Password for the administrative user, given in plain text. If no password is provided, it means that users can bind as <code>anonymous</code> . The password is encrypted in the configuration file.
<code>nsCheckLocalACI</code>	Reserved for advanced use only. Controls whether ACIs are evaluated on the database link as well as the remote data server. Takes the values <code>on</code> or <code>off</code> . Changes to this attribute occur only after the server has been restarted. The default value is <code>off</code> .
<code>nsProxiedAuthorization</code>	Reserved for advanced use only. Disables proxied authorization. A value of <code>off</code> means proxied authorization is disabled. The default value is <code>on</code> .
<code>nsActiveChainingComponents</code> [†]	Lists the components using chaining. A component is any functional unit in the server. The value of this attribute in the database link instance overrides the value in the global configuration attribute. To disable chaining on a particular database instance, use the value <code>none</code> . The default policy is not to allow chaining. For more information, see Section 3.1.1, “Chaining Component Operations” .
<code>nsReferralOnScopedSearch</code>	Controls whether referrals are returned by scoped searches. This attribute is for optimizing the directory because returning referrals in response to scoped searches is more efficient. Takes the values <code>on</code> or <code>off</code> . The default value is <code>off</code> .
<code>nsHopLimit</code>	Maximum number of times a request can be forwarded from one database link to another. The default value is <code>10</code> .

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

Table 3.4. Database Link Configuration Attributes

3.2.2.6. Database Link Configuration Example

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



1. Run `ldapmodify`¹ to add a database link to server A:

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

2. Specify the configuration information for the database link:

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

```
nsmultiplexorbinddn: cn=proxy admin,cn=config
nsmultiplexorcredentials: secret
cn: DBLink1

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

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

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

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

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



CAUTION

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

4. Add the following proxy authorization ACI to the

l=Zanzibar,ou=people,dc=example,dc=com entry on server B:

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


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



NOTE

When a user binds to a database link, the user's identity is sent to the remote server. Access controls are always evaluated on the remote server. For the user to modify or write data successfully to the remote server, set up the correct access controls on the remote server. For more information about how access controls are evaluated in the context of chained operations, see [Section 3.5, “Database Links and Access Control Evaluation”](#).

3.3. Chaining Using SSL

Database links can be configured to communicate with the remote server using SSL. Using SSL to chain involves the following steps:

1. Enable SSL on the remote server.
2. Specify the LDAP URL of the remote server in SSL format in the `nsFarmServerURL` attribute. For more information about this attribute, see [Section 3.2.2.3, “Providing an LDAP URL”](#). For example:

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

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

For more information on enabling SSL, see [Section 1.1, “Enabling SSL: Summary of Steps”](#).

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

3.4. Maintaining Database Links

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

- [Section 3.4.1, “Updating Remote Server Authentication Information”](#)
- [Section 3.4.2, “Deleting Database Links”](#)

3.4.1. Updating Remote Server Authentication Information

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

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

Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It takes the form `ldap://hostname:port/`.

5. Update the bind DN used by the database link to bind with the remote server by entering a new DN in the **Database link bind DN** field.
6. Update the password used by the database link to bind with the remote server by entering a new password in the **Database link password** field. Confirm the password by retyping it in the **Confirm database link password** field.

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

7. Click **Save**.

3.4.2. Deleting Database Links

To delete a database link, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation pane, locate the database link to delete, and select it.
3. From the **Object** menu, select **Delete**.

Alternatively, right-click the database link, and select **Delete** from the pop-up menu.

The **Deleting Database Link** confirmation dialog box is displayed.

4. Click **Yes** to confirm the deletion of the database link.

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

3.5. Database Links and Access Control Evaluation

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

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 may refer to remote users.
- ACIs must be located with any role definitions they use and with any users intended to have those roles.
- ACIs that refer to values of a user's entry (for example, `userattr` subject rules) will work if the user is remote.

Though access controls are always evaluated on the remote server, 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.

3.6. Advanced Feature: Tuning Database Link Performance

The following sections provide information on tuning the performance of database links through connection and thread management.

- [Section 3.6.1, “Managing Connections to the Remote Server”](#)
- [Section 3.6.2, “Detecting Errors During Normal Processing”](#)
- [Section 3.6.3, “Managing Threaded Operations”](#)

3.6.1. Managing Connections to the Remote Server

Each database link maintains a pool of connections to a remote server. The connections to optimize resources can be configured for the directory.

3.6.1.1. Managing Connections to the Remote Server Using the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder in the left pane and locate the database link to change. Click the **database link**, then click the **Limits and Controls** tab in the right navigation pane.
3. In the **Connection Management** section, make changes to any of the following fields:
 - *Maximum TCP connection(s)*. The maximum number of TCP connections that the database link establishes with the remote server. The default value is 3 connections.
 - *Bind timeout*. Amount of time, in seconds, before the database link's bind attempt times

out. The default value is 15 seconds.

- *Maximum binds per connection.* Maximum number of outstanding bind operations per TCP connection. The default value is 10 outstanding bind operations per connection.
- *Time out before abandon (sec).* Number of seconds before the server checks to see if a timed-out connection should be abandoned. The default value is 1 seconds.
- *Maximum LDAP connection(s).* Maximum number of LDAP connections that the database link establishes with the remote server. The default value is 10 connections.
- *Maximum bind retries.* Number of times a database link attempts to bind to the remote server. A value of 0 indicates that the database link will try to bind only once. The default value is 3 attempts.
- *Maximum operations per connection.* Maximum number of outstanding operations per LDAP connection. The default value is 2 operations per connection.
- *Connection lifetime (sec).* How long a connection made between the database link and remote server remains open. Connections between the database link and the remote server can be kept open for an unspecified time or closed after a specific period of time. It is faster to keep the connections open, but it uses more resources. For slow connections, it may be desirable to limit the connection time. A value of 0 indicates there is no limit. By default, the value is set to 0.

4. Click **Save**.

3.6.1.2. Managing Connections to the Remote Server from the Command-Line

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

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

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

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

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

The connection management attributes specified in this entry take precedence over the attributes specified in the `cn=default instance config` entry.

Attribute Name	Description
<code>nsOperationConnectionsLimit</code>	Maximum number of LDAP connections that

Attribute Name	Description
	the database link establishes with the remote server. The default value is 20 connections per database link instance.
nsBindConnectionsLimit	Maximum number of TCP connections that the database link establishes with the remote server. The default value is 3 connections.
nsConcurrentOperationsLimit	Maximum number of outstanding operations per LDAP connection. The default value is 2 operations per connection.
nsConcurrentBindLimit	Maximum number of outstanding bind operations per TCP connection. The default value is 10 outstanding bind operations.
nsBindRetryLimit	Number of times a database link attempts to bind to the remote server. A value of zero (0) indicates that the database link will try to bind only once. The default value is 3 attempts.
nsConnectionLife	Connection lifetime, in seconds. Connections between the database link and the remote server can be kept open for an unspecified time or closed after a specific period of time. It is faster to keep the connections open, but it uses more resources. For example, it may be wise to limit the connection time for a slow connection. A value of 0 indicates there is no limit. By default, the value is set to 0. When the value is 0 and there is a list of failover servers in the <i>nsFarmServerURL</i> attribute, the first server is never contacted after failover to the alternate server. The default value is 0 seconds.
nsBindTimeout	Amount of time, in seconds, before the bind attempt times out. The default value is 15 seconds.
nsAbandonedSearchCheckInterval	Number of seconds that pass before the server checks for abandoned operations. The default value is 1 seconds.

Table 3.5. Database Link Connection Management Attributes

For the list of database link configuration attributes, see [Table 3.4, “Database Link Configuration Attributes”](#).

3.6.2. Detecting Errors During Normal Processing

Protect server performance by detecting errors during the normal chaining operation between the database link and the remote server. The database link has two attributes —

nsMaxResponseDelay and *nsMaxTestResponseDelay* — which work together to determine if the remote server is no longer responding.

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

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

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

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

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

Table 3.6. Database Link Processing Error Detection Parameters

3.6.3. Managing Threaded Operations

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

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

While the database link waits for results from the remote server, it can process additional operations. By default, the number of threads used by the server is 30. However, when using database links, performance can be improved by increasing the number of threads available for processing operations. While the local CPU waits for a response from a remote server, it can process other operations rather than stand idle.

To change the number of threads used for processing operations, change the *nsslapd-thread number* global configuration attribute in the *cn=config* entry. Increasing the thread number can improve performance; the default thread number is 30. Restart the server after changing the thread count to apply the changes.

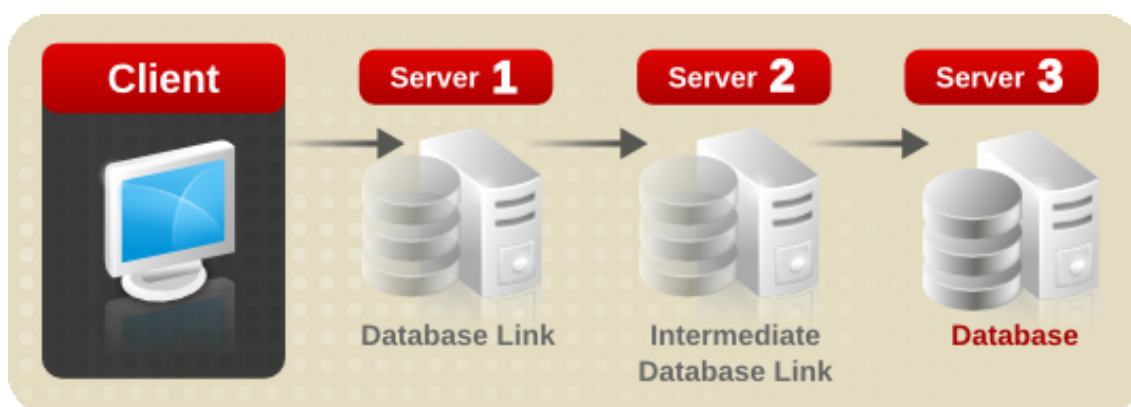
3.7. Advanced Feature: Configuring Cascading Chaining

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.

- [Section 3.7.1, “Overview of Cascading Chaining”](#)
- [Section 3.7.2, “Configuring Cascading Chaining Defaults Using the Console”](#)
- [Section 3.7.3, “Configuring Cascading Chaining Using the Console”](#)
- [Section 3.7.4, “Configuring Cascading Chaining from the Command-Line”](#)
- [Section 3.7.5, “Detecting Loops”](#)
- [Section 3.7.6, “Summary of Cascading Chaining Configuration Attributes”](#)
- [Section 3.7.7, “Cascading Chaining Configuration Example”](#)

3.7.1. Overview of Cascading Chaining

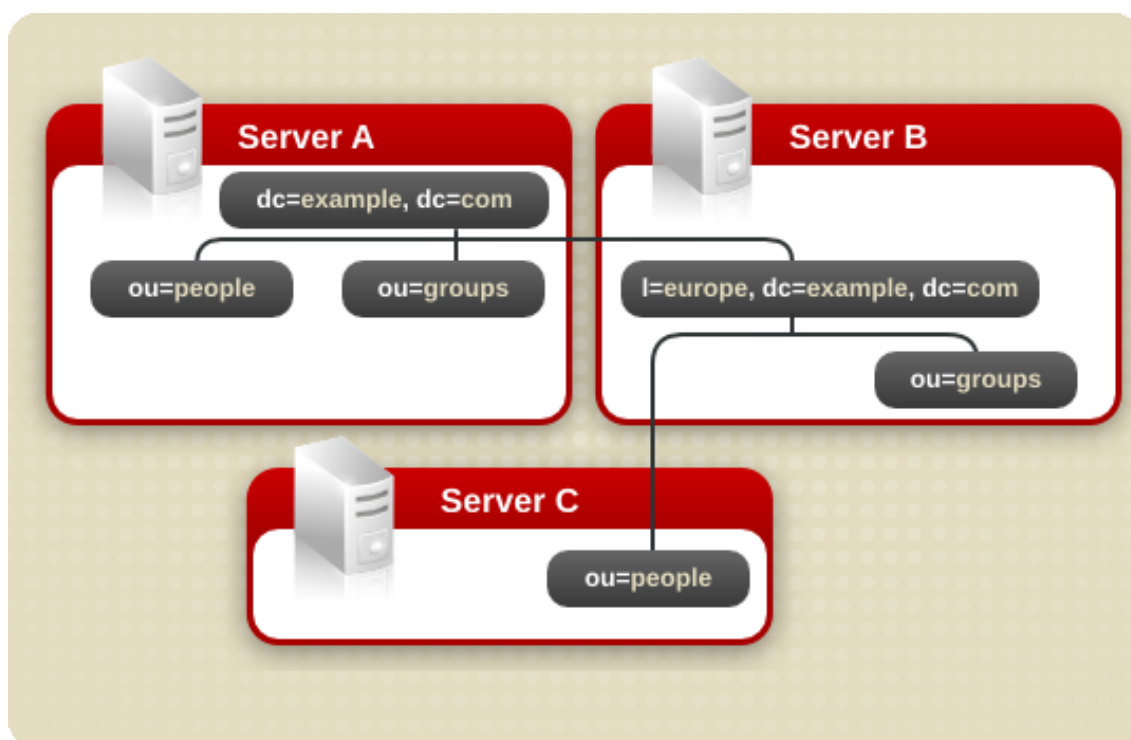
Cascading chaining occurs when more than one hop is required for the directory to process a client application's request. For example:



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

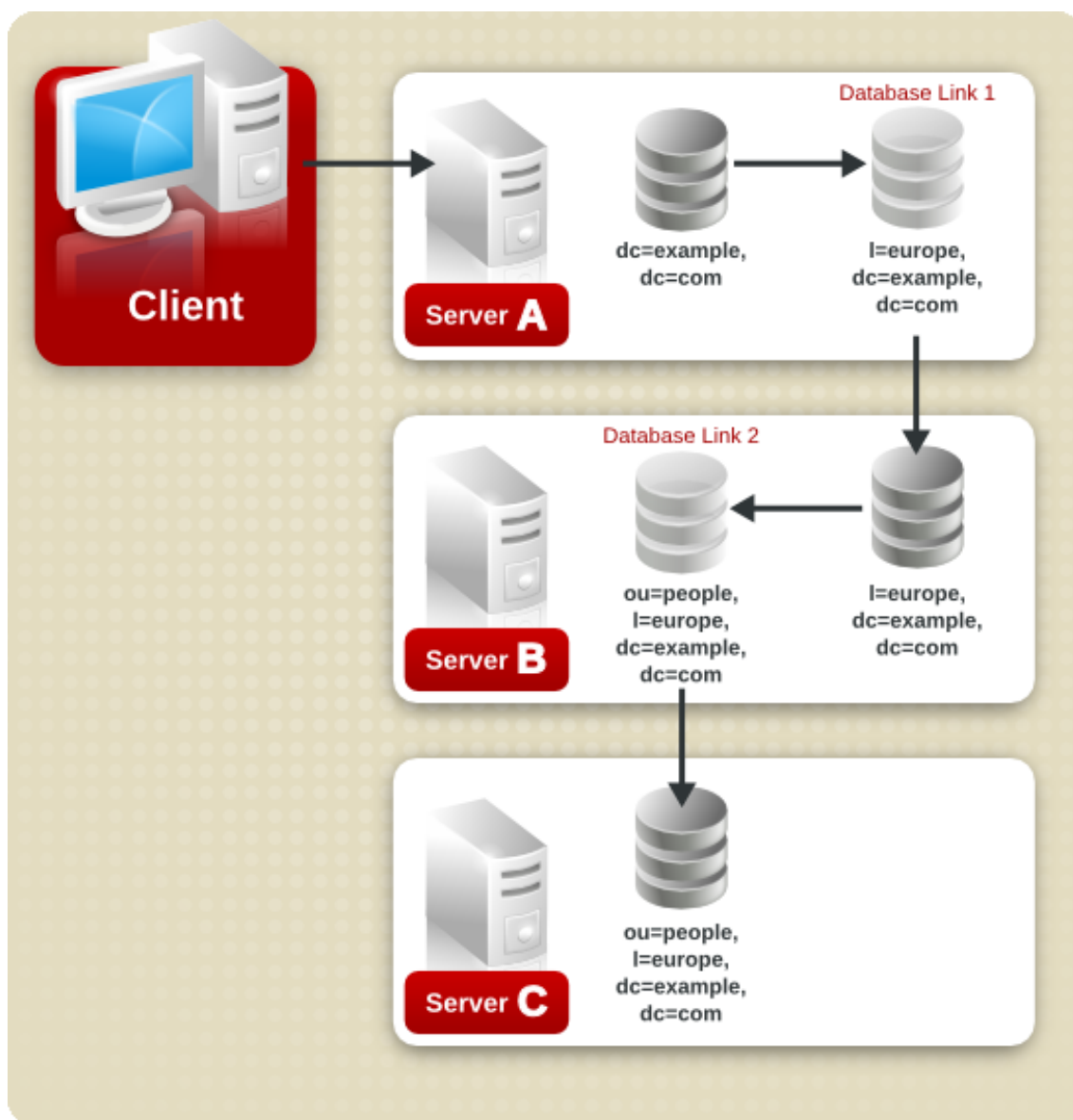
During a normal operation request, a client binds to the server, and then any ACIs applying to that client are evaluated. With cascading chaining, the client bind request is evaluated on server one, but the ACIs applying to the client are evaluated only after the request has been chained to the destination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

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



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

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



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

3.7.2. Configuring Cascading Chaining Defaults Using the Console

To set cascading chaining defaults for all database links in Directory Server, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder in the left pane, and click **Database Link Settings**. Click the **Default Creation Parameters** tab.
3. Select the **Check local ACI** checkbox to enable the evaluation of local ACIs on the intermediate database links involved in cascading chaining. Selecting this checkbox may require adding the appropriate local ACIs to a database on the servers that contain intermediate database links.
4. Enter the maximum number of times a database link can point to another database link in the **Maximum hops** field.

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

5. Click **Save**.



NOTE

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

3.7.3. Configuring Cascading Chaining Using the Console

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

1. In the Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** folder in the left pane, and locate the database link to include in a cascading chain. Click the database link, then click the **Limits and Controls** tab in the right navigation pane.
3. Select the **Check local ACI** checkbox to enable the evaluation of local ACIs on the intermediate database links involved in the cascading chain. Selecting this checkbox may require adding the appropriate local ACIs to the database link.
4. Enter the maximum number of times a database link can point to another database link in the **Maximum hops** field.

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

5. Click **Save**.

3.7.4. Configuring Cascading Chaining from the Command-Line

To configure a cascade of database links through the command-line, do the following:

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

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

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

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

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

```
nsTransmittedControls: 2.16.840.1.113730.3.4.12
```

The OID value represents the Proxy Authorization Control. For more information about chaining LDAP controls, see [Section 3.1.2, “Chaining LDAP Controls”](#).

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

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

- a. Create a database, if one does not already exist, on the server containing the intermediate database link. This database will contain the admin user entry and the ACI. For information about creating a database, see [Section 2.1, “Creating Databases”](#).
- b. Create an entry that corresponds to the administrative user in the database.
- c. Create an ACI for the administrative user that targets the appropriate suffix. This ensures

the administrator has access only to the suffix of the database link. For example:

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

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



CAUTION

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

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

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

```
nsCheckLocalACI: on
```

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

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

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

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

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

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

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

3.7.5. Detecting Loops

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

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

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

```
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

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

3.7.6. Summary of Cascading Chaining Configuration Attributes

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

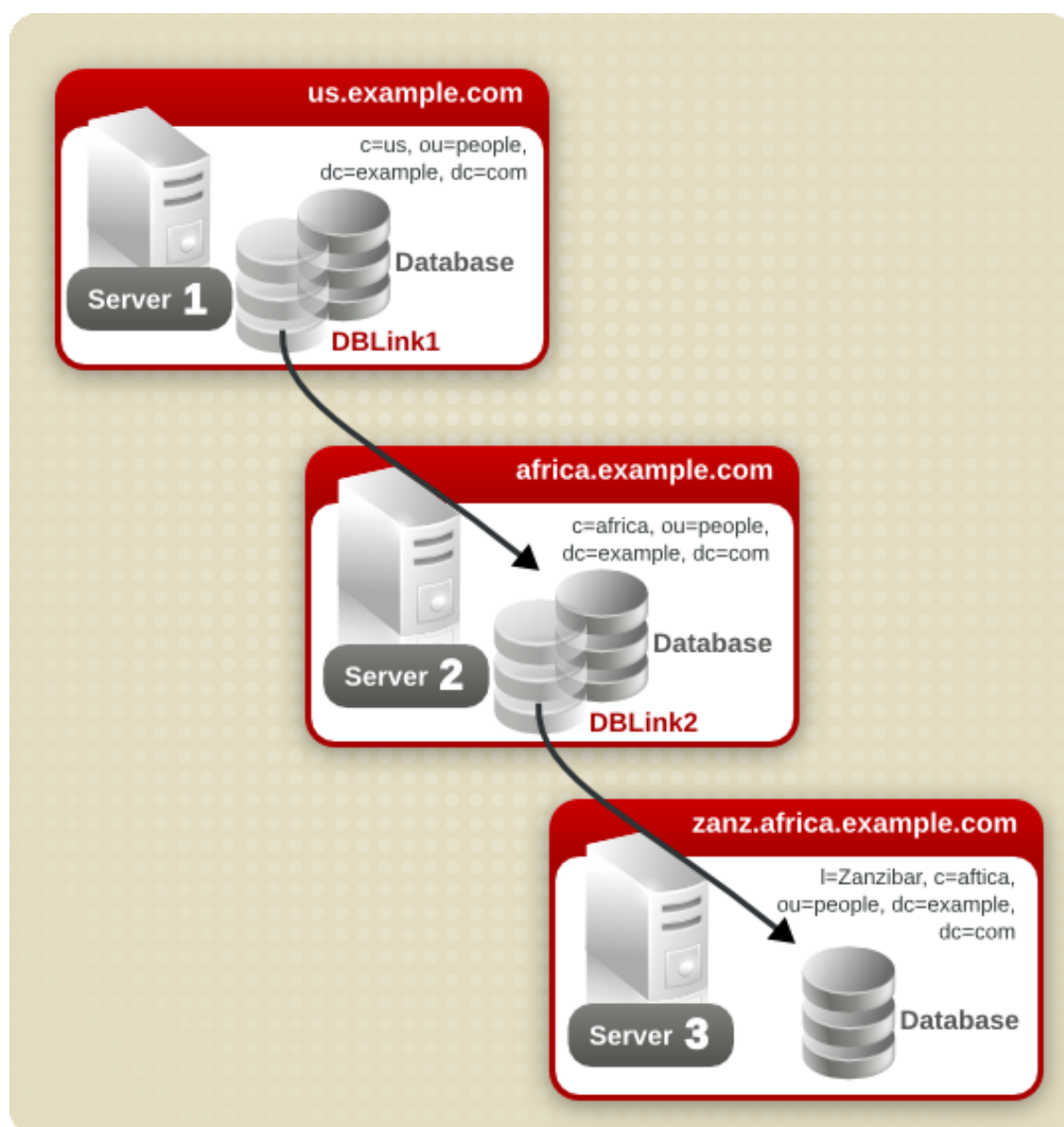
Attribute	Description
nsFarmServerURL	URL of the server containing the next database link in the cascading chain.
nsTransmittedControls	<div>Enter the following OIDs to the database links involved in the cascading chain:</div> <div><pre>nsTransmittedControls: 2.16.840.1.113730.3.4.12 nsTransmittedControls: 1.3.6.1.4.1.1466.29539.12</pre></div> <div>The first OID corresponds to the Proxy Authorization Control. The second OID corresponds to the Loop Detection Control.</div>

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

Table 3.7. Cascading Chaining Configuration Attributes

3.7.7. Cascading Chaining Configuration Example

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



- [Section 3.7.7.1, “Configuring Server One”](#)
- [Section 3.7.7.2, “Configuring Server Two”](#)
- [Section 3.7.7.3, “Configuring Server Three”](#)

3.7.7.1. Configuring Server One

1. Run `ldapmodify`¹:

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


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

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

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

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

3. To implement loop detection, to specify the OID of the loop detection control in the `nsTransmittedControl` attribute stored in `cn=config,cn=chaining database,cn=plugins,cn=config` entry on server one.

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

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

3.7.7.2. Configuring Server Two

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

follows:

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



CAUTION

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

2. Configure the database link, `DBLink2`, on server two, using `ldapmodify`:

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

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

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

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

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

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

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

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

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

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



NOTE

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

- a. Add the local proxy authorization ACI to the `c=africa,ou=people,dc=example,dc=com` entry:

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

- b. Then add the local client ACI that will allow the client operation to succeed on server two, given that ACI checking is turned on. This ACI is the same as the ACI created on the destination server to provide access to the

`l=Zanzibar,c=africa,ou=people,dc=example,dc=com` branch. All users within `c=us,ou=people,dc=example,dc=com` may need to have update access to the entries in `l=Zanzibar,c=africa,ou=people,dc=example,dc=com` on server three. Create the following ACI on server two on the `c=africa,ou=people,dc=example,dc=com` suffix to allow this:

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

This ACI allows clients that have a UID in `c=us,ou=people,dc=example,dc=com` on server one to perform any type of operation on the

`l=Zanzibar,c=africa,ou=people,dc=example,dc=com` suffix tree on server three. If there are users on server two under a different suffix that will require additional rights on server three, it may be necessary to add additional client ACIs on server two.

3.7.7.3. Configuring Server Three

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

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

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

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

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

3. Create a local client ACI on the `l=Zanzibar,ou=people,dc=example,dc=com` subtree that

corresponds to the original client application. Use the same ACI as the one created for the client on server two:

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

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

4. 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 4.1, “Starting the Server in Referral Mode”](#)
- [Section 4.2.1, “Setting a Default Referral Using the Console”](#)
- [Section 4.3, “Creating Smart Referrals”](#)
- [Section 4.4, “Creating Suffix Referrals”](#)

4.1. Starting the Server in Referral Mode

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

Run `nsslapd` with the `refer` option.

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

- `/usr/lib/dirsrv/slapd-instance_name` is the directory where the Directory Server configuration files are. This is the default location on Red Hat Enterprise Linux 5 i386; for the

location on other platforms, see [Section 1, “Directory Server File Locations”](#).

- *port* is the optional port number of the Directory Server to start in referral mode.
- *referral_url* is the referral returned to clients. The format of an LDAP URL is covered in [Appendix C, LDAP URLs](#).

4.2. Setting Default Referrals

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

4.2.1. Setting a Default Referral Using the Console

Set a default referral to the directory, as follows:

1. In the Directory Server Console, select the **Configuration** tab.
2. Select the top entry in the navigation tree in the left pane.
3. Select the **Settings** tab in the right pane.
4. Enter an LDAP URL in the **Referrals to** text box.

For example:

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

Enter multiple referral URLs separated by spaces and in quotes, as follows:

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

For more information about LDAP URLs, see [Appendix C, LDAP URLs](#).

5. Click **OK**.

4.2.2. Setting a Default Referral from the Command-Line

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

1. Run the `ldapmodify` utility:¹

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

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

2. Add the default referral to the `dir2.example.com` server:

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

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

4.3. 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>.

4.3.1. Creating Smart Referrals Using the Directory Server Console

To configure smart referrals, do the following:

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

The **Edit Smart Referrals** dialog box opens.

4. Select the **Enable Smart Referral** option to define smart referrals for the selected entry. (Unchecking the option removes all smart referrals from the entry and deletes the `referral` object class from the entry.)

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

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

optional_dn is the explicit DN for the server to return to the requesting client application. For example, this LDAP URL references John Doe's entry:

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

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

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

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

6. To allow a referral to be followed with different authentication, click **Authentication**, and specify the appropriate DN and password. Keep in mind that this authentication remains valid only until the Console is closed; then it's reset to the same authentication used to log into the Console.
7. The **Smart Referral List** lists the referrals currently in place for the selected entry. The entire list of referrals is returned to client applications in response to a request with the **Return Referrals for All Operations** or **Return Referrals for Update Operations** options in the **Suffix Settings** tab, which is available under the **Configuration** tab.

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

8. Click **OK**.

4.3.2. Creating Smart Referrals from the Command Line

Use the `ldapmodify` command-line utility¹ to create smart referrals from the command-line.

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

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

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


**NOTE**

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

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

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

Use the `-M` option with `ldapmodify` when there is already a referral in the DN path. For information about the `ldapmodify` utility, see the *Directory Server Configuration, Command, and File Reference*.

4.4. 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.

**CAUTION**

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

4.4.1. Creating Suffix Referrals Using the Console

To create a suffix referral using the Console, do the following:

1. Select the **Configuration** tab.
2. Under **Data** in the left pane, click the suffix to which to add a referral.

3. In the **Suffix Settings** tab, select one of the following radio buttons:

- **Return Referrals for all Operations.** This means that a referral will be returned when this suffix receives any request from a client application.
- **Return Referrals for Update Operations.** This means a referral will be returned when this suffix receives an update request from a client application. This option is used to redirect update and write requests made by client applications to a read-only database.

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

For more information about the structure of LDAP URLs, see [Appendix C, LDAP URLs](#).

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

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

6. Click **Save**.

4.4.2. Creating Suffix Referrals from the Command-Line

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

1. Run `ldapmodify`.¹ For example:

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

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

2. Add a suffix referral to the `ou=people,dc=example,dc=com` root suffix, as follows:

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

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

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

an update request to a suffix set to `referral on update`, the client receives a referral.

For more information about the suffix configuration attributes, refer to [Table 3.1, “Suffix Attributes”](#).

Populating Directory Databases

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

1. Importing Data

Directory Server provides three methods for importing data:

- *Import from the Directory Server Console.* Use the Directory Server Console to append data to all of the databases, including database links.
- *Initialize databases.* The Directory Server Console can import data to one database; this method overwrites any data contained by the database.
- *Importing data from the command-line.* Directory Server provides command-line utilities to import data.



NOTE

The LDIF files used for import operations must use UTF-8 character set encoding. Import operations do not convert data from local character set encoding to UTF-8 character set encoding.

Table 4.1, “*Import Method Comparison*” describes the differences between an import and initializing databases.

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

Table 4.1. Import Method Comparison

The following sections describe importing data:

- [Section 1.1, “Importing a Database from the Console”](#)
- [Section 1.2, “Initializing a Database from the Console”](#)
- [Section 1.3, “Importing from the Command-Line”](#)



CAUTION

All imported LDIF files must also contain the root suffix.

1.1. Importing a Database from the Console

When performing an import operation from the Directory Server Console, an `ldapmodify` operation is executed to append data, as well as to modify and delete entries. The operation is performed on all of the databases managed by the Directory Server and on remote databases to which the Directory Server has a configured database link.

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

To import data from the Directory Server Console, do the following:

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

Alternatively, import by going to the **Configuration** tab and selecting **Import** from the **Console** menu.

2. In the **Import Database** dialog box, enter the full path to the LDIF file to import in the **LDIF file** field, or click **Browse** to select the file to import.

If the Console is running on a machine remote to the directory, the field name appears as **LDIF file (on the machine running the Console)**. When browsing for a file, you are not browsing the current directory for the Directory Server host, but the filesystem of the machine running the Console.

3. In the **Options** box, select one or both of the following options:
 - **Add Only**. The LDIF file may contain modify and delete instructions in addition to the default add instructions. For the server to ignore operations other than add, select the **Add only** checkbox.
 - **Continue on Error**. Select the **Continue on error** checkbox for the server to continue with the import even if errors occur. For example, use this option to import an LDIF file that

contains some entries that already exist in the database in addition to new ones. The server notes existing entries in the rejects file while adding all new entries.

4. In the **File for Rejects** field, enter the full path to the file in which the server is to record all entries it cannot import, or click **Browse** to select the file which will contain the rejects.

A reject is an entry which cannot be imported into the database; for example, the server cannot import an entry that already exists in the database or an entry that has no parent object. The Console will write the error message sent by the server to the rejects file.

Leaving this field blank means the server will not record rejected entries.

5. Click **OK**.

The server performs the import and also creates indexes.



NOTE

Trailing spaces are dropped during a remote Console import but are preserved during both local Console or `ldif2db` import operations.

1.2. Initializing a Database from the Console

The existing data in a database can be overwritten by initializing databases.

You must be logged in as the `Directory Manager` in order to initialize a database because an LDIF file that contains a root entry cannot be imported into a database except as the Directory Manager (root DN). Only the Directory Manager has access to the root entry, such as `dc=example,dc=com`.



CAUTION

When initializing databases from an LDIF file, be careful not to overwrite the `o=NetscapeRoot` suffix unless you are restoring data. Otherwise, initializing the database deletes information and may require re-installing the Directory Server.

To initialize a database using the Directory Server Console, do the following:

1. Select the **Configuration** tab.
2. Expand the **Data** tree in the left navigation pane. Expand the suffix of the database to initialize, then click the database itself.

3. Right-click the database, and select **Initialize Database**.

Alternatively, select **Initialize Database** from the **Object** menu.

4. In the **LDIF file** field, enter the full path to the LDIF file to import, or click **Browse**.

5. If the Console is running from a machine local to the file being imported, click **OK** and proceed with the import immediately. If the Console is running from a machine remote to the server containing the LDIF file, select one of the following options, then click **OK**:

- *From local machine*. Indicates that the LDIF file is located on the local machine.
- *From server machine*. Indicates that the LDIF file is located on a remote server.

The default LDIF directory is `/var/lib/dirsrv/slapd-instance_name/ldif`.¹

1.3. Importing from the Command-Line

There are three methods for importing data through the command-line:

- *Using `ldif2db`*. This import method overwrites the contents of the database and requires the server to be stopped; see [Section 1.3.1, “Importing Using the `ldif2db` Command-Line Script](#)”.
- *Using `ldif2db.pl`*. This import method overwrites the contents of the database while the server is still running; see [Section 1.3.2, “Importing Using the `ldif2db.pl` Perl Script](#)”.
- *Using `ldif2ldap`*. This method appends the LDIF file through LDAP. This method is useful to append data to all of the databases; see [Section 1.3.3, “Importing Using the `ldif2ldap` Command-Line Script](#)”.



NOTE

To import a database that has been encrypted, use the `-E` option with the script. See [Section 2.3.5, “Exporting and Importing an Encrypted Database](#)” for more information.

1.3.1. Importing Using the `ldif2db` Command-Line Script

The `ldif2db` script overwrites the data in the specified database. Also, the script requires that the Directory Server be stopped when the import begins.

By default, the script first saves and then merges any existing `o=NetscapeRoot` configuration information with the `o=NetscapeRoot` configuration information in the files being imported.

¹ This is the location for Red Hat Enterprise Linux. File locations for other platforms are listed in [Section 1, “Directory Server File Locations](#)”.

**CAUTION**

This script overwrites the data in the database.

To import LDIF, do the following:

1. Stop the server.²

```
service dirsrv stop instance
```

2. Open the Directory Server instance directory.

```
cd /usr/lib/dirsrv/slapd-instance_name
```

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

```
ldif2db -n Databasel -i /var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif  
-i /var/lib/dirsrv/slapd-instance_name/ldif/demo2.ldif
```

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

**CAUTION**

If the database specified in the `-n` option does not correspond with the suffix contained by the LDIF file, all of the data contained by the database is deleted, and the import fails. Make sure that the database name is not misspelled.

Option	Description
<code>-i</code>	Specifies the full path name of the LDIF files to be imported. This option is required. To import more than one LDIF file at a time, use multiple <code>-i</code> arguments. When multiple files are imported, the server imports the LDIF files in the order which they are specified from the command-line.
<code>-n</code>	Specifies the name of the database to which to import the data.


² The command to start and stop the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, “Starting and Stopping Servers”](#).

Table 4.2. Idif2db Parameters

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

1.3.2. Importing Using the Idif2db.pl Perl Script

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



CAUTION

This script overwrites the data in the database.


1. Open the Directory Server instance directory.

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Run the `ldif2db` script.

```
ldif2db -D "cn=Directory Manager" -w secretpwd  
-i /var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif -n Database1
```

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.



NOTE

You do not need `root` privileges to run the script, but you must authenticate as the Directory Manager.

Option	Description
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
-i	Specifies the LDIF files to be imported. This option is required. To important multiple LDIF

Option	Description
	files at a time, use multiple <code>-i</code> arguments. When multiple files are imported, the server imports the LDIF files in the order they are specified in the command-line.
<code>-n</code>	Specifies the name of the database to which to import the data.

Table 4.3. ldif2db Options

1.3.3. Importing Using the ldif2ldap Command-Line Script

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

To import LDIF using `ldif2ldap`, do the following:

1. Open the Directory Server instance directory:

```
cd /usr/lib/dirsrv/slapd-instance_name
```

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

```
ldif2ldap "cn=Directory Manager" secretpwd  
/var/lib/dirsrv/slapd-instance_name/ldif/demo.ldif
```

The `ldif2ldap` script requires the DN of the administrative user, the password of the administrative user, and the absolute path and filename of the LDIF files to be imported.

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

2. Exporting Data

LDAP Data Interchange Format (LDIF) files are used to export database entries from the Directory Server databases. LDIF is a standard format described in RFC 2849, *The LDAP Data Interchange Format (LDIF) - Technical Specification*.

Exporting data can be useful for the following:

- Backing up the data in the database.

- Copying data to another Directory Server.
- Exporting data to another application.
- Repopulating databases after a change to the directory topology.

For example, if a directory contains one database, and its contents are split into two databases, then the two new databases receive their data by exporting the contents of the old databases and importing it into the two new databases, as illustrated in [Figure 4.1, “Splitting a Database Contents into Two Databases”](#).



NOTE

The export operations do not export the configuration information (`cn=config`), schema information (`cn=schema`), or monitoring information (`cn=monitor`).

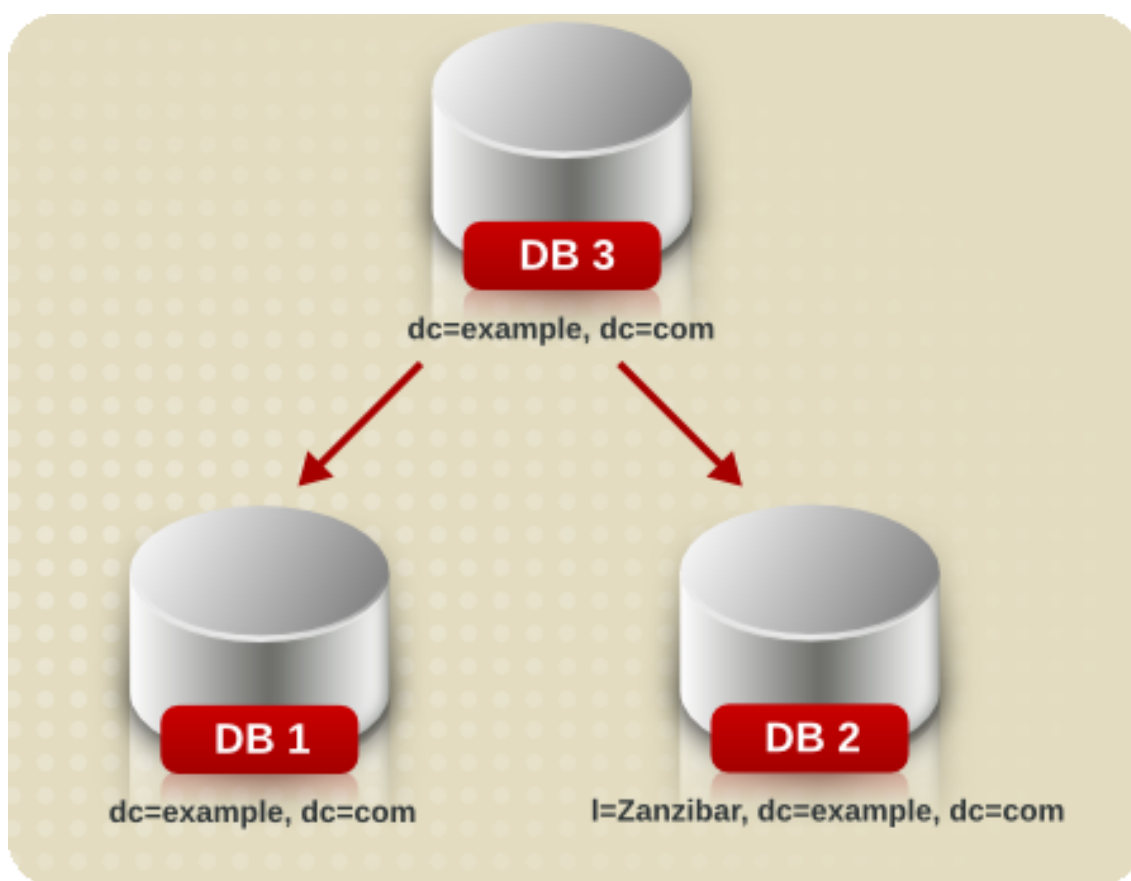


Figure 4.1. Splitting a Database Contents into Two Databases

The Directory Server Console or command-line utilities can be used to export data.

- [Section 2.1, “Exporting Directory Data to LDIF Using the Console”](#)
- [Section 2.2, “Exporting a Single Database to LDIF Using the Console”](#)
- [Section 2.3, “Exporting to LDIF from the Command-Line”](#)



CAUTION

Do not stop the server during an export operation.

2.1. Exporting Directory Data to LDIF Using the Console

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

Export directory data to LDIF from the Directory Server Console while the server is running, and do the following:

1. Select the **Tasks** tab. Scroll to the bottom of the screen, and click **Export Database(s)**.

Alternatively, select the **Configuration** tab and click the **Export from the Console** menu.

The **Export Database** dialog box opens.

2. Enter the full path and filename of the LDIF file in the **LDIF File** field, or click **Browse** to locate the file.

Browse is not enabled if the Console is running on a remote server. When the **Browse** button is not enabled, the file is stored in the default directory,

`/var/lib/dirsrv/slaped-instance_name/ldif.`¹

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

- Select **To local machine** to export the data to an LDIF file on the machine from which the Console is running.
- Select **To server machine** to export to an LDIF file located on the server's machine.

4. To export the whole directory, select the **Entire database** radio button.

To export only a single subtree of the suffix contained by the database, select the **Subtree** radio button, and then enter the name of the suffix in the **Subtree** text box. This option exports a subtree that is contained by more than one database.

Alternatively, click **Browse** to select a suffix or subtree.

5. Click **OK** to export the file.

2.2. Exporting a Single Database to LDIF Using the Console

It is also possible to export a single database to LDIF. Do the following while the server is running:

1. Select the **Configuration** tab.
2. Expand the **Data** tree in the left navigation pane. Expand the suffix, and select the database under the suffix.
3. Right-click the database, and select **Export Database**.

Alternatively, select **Export Database** from the **Object** menu.

The **Export Partition** dialog box opens.

4. In the **LDIF file** field, enter the full path to the LDIF file, or click **Browse**.

When the **Browse** button is not enabled, the file is stored in the default directory, `/var/lib/dirsrv/slapd-instance_name/ldif`.¹

5. Click **OK** to export the file.

2.3. Exporting to LDIF from the Command-Line

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



NOTE

To export a database that has been encrypted, you must use the `-E` option with the script. See [Section 2.3.5, “Exporting and Importing an Encrypted Database”](#) for more information.

To export to LDIF from the command-line, do the following:

1. Open the Directory Server instance directory:

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Run the db2ldif command-line script.

```
db2ldif -n database1 -a /export/output.ldif
```

This exports the database contents to `/export/output.ldif`. If the `-a` option is not specified, then the database information is exported to `/var/lib/dirsrv/slapd-instance_name/ldif/instance_name-database1-date.ldif`. For example:

```
db2ldif -n database1
```

It is also possible to specify which suffixes to export, using the `-s` option. For example:

```
db2ldif -s "dc=example,dc=com"
```

The LDIF file in this case would be

`/var/lib/dirsrv/slapd-instance_name/ldif/instance_name-example-2007_04_30_112718.ldif`, using the name of the suffix rather than the database.

If the suffix specified is a root suffix, such as `dc=example,dc=com`, then it is not necessary to specify the database or to use the `-n` option.

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

Option	Description
-n	Specifies the name of the database from which the file is being exported.
-s	Specifies the suffix or suffixes to include in the export. If the suffix is a root suffix, such as <code>dc=example,dc=com</code> , then the <code>-n</code> option is not required. There can be multiple <code>-s</code> arguments.
-a	Defines the output file to which Directory Server exports the LDIF. This file must be an absolute path. If the <code>-a</code> option is not given, the output ldif is stored in the <code>/var/lib/dirsrv/slapd-<i>instance_name</i>/ldif</code> directory and is automatically named <code><i>serverID</i>-database-YYYY_MM_DD_hhmmxx.ldif</code> with the <code>-n</code> option or <code><i>serverID-firstsuffixvalue</i>-YYYY_MM_DD_hhmmxx.ldif</code>

Option	Description
	with the <code>-s</code> option.

Table 4.4. db2ldif Options

3. Backing up and Restoring Data

Databases can be backed up and restored using the Directory Server Console or a command-line script.

- [Section 3.1, “Backing up All Databases”](#)
- [Section 3.2, “Backing up the `dse.ldif` Configuration File”](#)
- [Section 3.3, “Restoring All Databases”](#)
- [Section 3.4, “Restoring a Single Database”](#)
- [Section 3.5, “Restoring Databases That Include Replicated Entries”](#)
- [Section 3.6, “Restoring the `dse.ldif` Configuration File”](#)



CAUTION

Do not stop the server during a backup or restore operation.

3.1. Backing up All Databases

The following procedures describe backing up all of the databases in the directory using the Directory Server Console and from the command-line.



NOTE

These backup methods cannot be used to back up the data contained by databases on a remote server that are chained using database links.

3.1.1. Backing up All Databases from the Server Console

When backing up databases from the Directory Server Console, the server copies all of the database contents and associated index files to a backup location. A backup can be performed

while the server is running.

To back up databases from the Directory Server Console, do the following:

1. Select the **Tasks** tab.
2. Click **Back Up Directory Server**.

The **Backup Directory** dialog box opens.

3. Enter the full path of the directory to store the backup file in the **Directory** text box, or click **Use default**, and the server provides a name for the backup directory.

If the Console is running on the same machine as the directory, click **Browse** to select a local directory.

With the default location, the backup files are placed in `/var/lib/dirsrv/slapd-instance_name/bak`.¹ By default, the backup directory name contains the name of the server instance and the time and date the backup was created (*instance_name-YYYY_MM_DD_hhmmss*).

4. Click **OK** to create the backup.

3.1.2. Backing up All Databases from the Command-Line

Databases can be backed up from the command-line using the `db2bak` command-line script. This script works when the server is running or when the server is stopped.

Configuration information *cannot* be backed up using this backup method. For information on backing up the configuration information, refer to [Section 3.1.2, “Backing up All Databases from the Command-Line”](#).

To back up the directory from the command-line using the `db2bak` script, do the following:

1. Open the Directory Server instance directory:

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Run the `db2bak` command-line script.

```
db2bak  
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2007_04_30_16_27_56
```

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

The backup directory where the server saves the backed up databases can be specified with the script. If a directory is not specified, the backup file is stored in `/var/lib/dirsrv/slapd-instance_name/bak`.¹ By default, the backup directory is named with the Directory Server instance name and the date of the backup (`serverID-YYYY_MM_DD_hhmmss`).

3.2. Backing up the `dse.ldif` Configuration File

Directory Server automatically backs up the `dse.ldif` configuration file. When the Directory Server is started, the directory creates a backup of the `dse.ldif` file automatically in a file named `dse.ldif.startOK` in the `/etc/dirsrv/slapd-instance_name` directory.

When the `dse.ldif` file is modified, the file is first backed up to a file called `dse.ldif.bak` in the `/etc/dirsrv/slapd-instance_name` directory before the directory writes the modifications to the `dse.ldif` file.

3.3. Restoring All Databases

The following procedures describe restoring all of the databases in the directory using the Directory Server Console and from the command-line.



NOTE

While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

3.3.1. Restoring All Databases from the Console

If the databases become corrupted, restore data from a previously generated backup using the Directory Server Console. This process consists of stopping the server and then copying the databases and associated index files from the backup location to the database directory.



CAUTION

Restoring databases overwrites any existing database files.

To restore databases from a previously created backup, do the following:

1. In the Directory Server Console, select the **Tasks** tab.
2. Click **Restore Directory Server**.

The **Restore Directory** dialog box is displayed.

3. Select the backup from the **Available Backups** list, or enter the full path to a valid backup in the **Directory** text box.

The **Available Backups** list shows all backups located in the default directory, `/var/lib/dirsrv/slapd-instance_name/bak/backup_directory`. ¹*backup_directory* is the directory of the most recent backup, in the form *serverID-YYYY_MM_DD_hhmmss*.

4. Click **OK**.

3.3.2. Restoring Your Database from the Command-Line

Restore databases from the command-line by using the following scripts:

- Using the `bak2db` command-line script. This script requires the server to be shut down.
- Using the `bak2db.pl` Perl script. This script works while the server is running.

3.3.2.1. Using the bak2db Command-Line Script

To restore the directory from the command-line, do the following:

1. If the Directory Server is running, stop it:²

```
service dirsrv stop instance
```

2. Open the Directory Server instance directory:

```
cd /usr/lib/dirsrv/slapd-instance_name
```

3. Run the `bak2db` command-line script. The `bak2db` script requires the full path and name of the input file.

```
bak2db  
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2007_04_30_11_48_30
```

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

3.3.2.2. Using bak2db.pl Perl Script

To restore the directory from the command-line, do the following while the server is running:

1. Open the Directory Server instance directory:

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Run the bak2db.pl Perl script.

```
bak2db.pl -D "cn=Directory Manager" -w secret  
-a  
/var/lib/dirsrv/slapd-instance_name/bak/instance_name-2007_04_30_11_48_30
```

For more information on using this Perl script, see the *Directory Server Configuration, Command, and File Reference*.

Option	Description
-a	Defines the full path and name of the input file.
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.

3.4. Restoring a Single Database

It is possible to restore a single database through the command-line, but not in the Directory Server Console. To restore a single database, do the following:

1. Stop the Directory Server if it is running.²

```
service dirsrv stop instance
```

2. Restore the backend from the `/var/lib/dirsrv/slapd-instance_name/bak` archives with the bak2db script, using the `-n` parameter to specify the database name. For example:

```
bak2db /var/lib/dirsrv/slapd-instance_name/bak/backup_file -n userRoot
```

3. Restart the Directory Server.

```
service dirsrv start instance
```

**NOTE**

If the Directory Server fails to start, remove the database transaction log files in `/var/lib/dirsrv/slapd-instance_name/db/log.###`, then retry starting the server.

3.5. Restoring Databases That Include Replicated Entries

If a database that supplies entries to other servers is restored, then you must reinitialize all of the servers that receive updates from the restored database (for example, consumer servers, hub servers, and, in multi-master replication environments, other supplier servers). The changelog associated with the restored database will be erased during the restore operation. A message will be logged to the supplier servers' log files indicating that reinitialization is required. If a database containing data received from a supplier server is restored, then one of two situations can occur:

- Changelog entries have not yet expired on the supplier server.

If the supplier's changelog has not expired since the database backup was taken, then restore the local consumer and continue with normal operations. This situation occurs only if the backup was taken within a period of time that is shorter than the value set for the maximum changelog age attribute, `nsslapd-changelogmaxage`, in the `cn=changelog5,cn=config` entry. For more information about this option, see the *Directory Server Configuration, 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, refer to [Section 10, “Initializing Consumers”](#).

For information on managing replication, see [Chapter 8, Managing Replication](#).

3.6. Restoring the `dse.ldif` Configuration File

The directory creates two backup copies of the `dse.ldif` file in the `/etc/dirsrv/slapd-instance_name` directory. The `dse.ldif.startOK` file records a copy of the `dse.ldif` file at server start up. The `dse.ldif.bak` file contains a backup of the most recent changes to the `dse.ldif` file. Use the version with the most recent changes to restore the directory.

To restore the `dse.ldif` configuration file, do the following:

1. Stop the server.²

```
service dirsrv stop instance
```

2. Restore the database as outlined in [Section 3.4, “Restoring a Single Database”](#) to copy the backup copy of the `dse.ldif` file into the directory.

3. Restart the server.

```
service dirsrv restart instance
```

Managing Entries with Roles, Class of Service, and Views

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.

1. Using Roles

Roles are a new entry grouping mechanism that unify the static and dynamic groups described in the previous sections. Roles are designed to be more efficient and easier to use for applications. For example, an application can 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.

This section contains the following topics:

- [Section 1.1, “About Roles”](#)
- [Section 1.2, “Managing Roles Using the Console”](#)
- [Section 1.3, “Managing Roles Using the Command-Line”](#)
- [Section 1.4, “Using Roles Securely”](#)

1.1. About Roles

Roles unify the static and dynamic group concept supported by previous versions of Directory Server.

Roles can be used to organize users in number of different ways:

- To enumerate the members of a role.

Having an enumerated list of role members can be useful for resolving queries for role members quickly.

- To determine whether a given entry possesses a particular role.

Knowing the roles possessed by an entry can help determine whether the entry possesses the target role.

- To enumerate all the roles possessed by a given entry.

- To assign a particular role to a given entry.
- To remove a particular role from a given entry.

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.

However, evaluating roles is more resource-intensive because the server does the work for the client application. With roles, the client application can check role membership by searching the *nsRole* attribute. The *nsRole* attribute is a computed attribute, which identifies to which roles an entry belongs; the *nsRole* attribute is not stored with the entry itself. From the client application point of view, the method for checking membership is uniform and is performed on the server side.



NOTE

The *nsRole* attribute is an operational attribute. In LDAP, operational attributes must be requested explicitly in the search attributes list; they are not returned by default with the regular attributes in the schema of the entry. For example, this `ldapsearch` command returns the list of roles of which `uid=scarter` is a member, in addition to the regular attributes for the entry:

```
ldapsearch ... args ... "(uid=scarter)" \* nsRole
```

Be sure to use the *nsRole* attribute, not the *nsRoleDN* attribute, to evaluate role membership.

The Console will automatically show the roles.

Each role has *members*, or 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 are said to possess the role.
- *Nested roles* are roles that contain other roles.

The concept of activating/inactivating roles allows entire groups of entries to be activated or inactivated in just one operation. That is, the members of a role can be temporarily disabled by

inactivating the role to which they belong.

When a role is inactivated, it does not mean that the user cannot bind to the server using that role entry. The meaning of an inactivated role is that the user cannot bind to the server using any of the entries that belong to that role; the entries that belong to an inactivated role will have the `nsAccountLock` attribute set to `true`.

In the case of the nested role, an inactivated nested role means that a user cannot bind to the server using an entry that belongs to a role that is a member of the nested role. All the entries that belong to a role that directly or indirectly are members of the nested role (one may have several levels of nested roles) will have `nsAccountLock` set to `true`.



NOTE

The `nsAccountLock` attribute is an operational attribute and must be explicitly requested in the search command in the list of search attributes. For example:

```
ldapsearch ... args ... "(uid=scarter)" \* nsAccountLock
```

The Console will automatically show the active/inactive status of entries.

1.2. Managing Roles Using the Console

This section contains the following procedures for creating and modifying roles:

- [Section 1.2.1, “Creating a Managed Role”](#)
- [Section 1.2.2, “Creating a Filtered Role”](#)
- [Section 1.2.3, “Creating a Nested Role”](#)
- [Section 1.2.4, “Viewing and Editing an Entry's Roles”](#)
- [Section 1.2.5, “Modifying a Role Entry”](#)
- [Section 1.2.6, “Making a Role Inactive”](#)
- [Section 1.2.7, “Reactivating a Role”](#)
- [Section 1.2.8, “Deleting a Role”](#)

When a role is created, determine whether a user can add themselves or remove themselves from the role. See [Section 1.4, “Using Roles Securely”](#) for more information about roles and access control.

1.2.1. 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.

To create and add members to a managed role, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new role.
3. Go to the **Object** menu, and select **New > Role**.

Alternatively, right-click the entry and select **New > Role**.

The **Create New Role** dialog box is displayed.

4. Click **General** in the left pane. Type a name for the new role in the **Role Name** field.

The role name is required.

5. Enter a description of the new role in the **Description** field.

6. Click **Members** in the left pane.

A search dialog box appears briefly.

7. In the right pane, select Managed Role. Click **Add** to add new entries to the list of members.

The standard **Search users and groups** dialog box appears.

8. In the **Search** drop-down list, select **Users** from the **Search** drop-down list, then click **Search**. Select one of the entries returned, and click **OK**.

9. After adding all of the entries, click **OK**.

The new role appears in the right pane.



NOTE

The *nsRoleDN* attribute is an operational attribute and must be explicitly requested in the search command in the list of search attributes. For example:

```
ldapsearch ... args ... "(uid=scarter)" \* nsRole nsRoleDN
```

The Console will automatically show the *nsRoleDN* attribute.

1.2.2. Creating a Filtered Role

Entries are assigned to a filtered role depending upon a particular attribute contained by each entry. The role definition specifies an LDAP filter for the target attributes. Entries that match the filter possess (are members of) the role.

To create and add members to a filtered role, do the following:

1. Follow the steps of [Section 1.2.1, "Creating a Managed Role"](#).

2. Click **Members** in the left pane.

A search dialog box appears briefly.

3. In the right pane, select **Filtered Role**.

4. Enter an LDAP filter in the text field, or click **Construct** to be guided through the construction of an LDAP filter.

5. The **Construct** opens the standard LDAP URL construction dialog. Ignore the fields for **LDAP Server Host**, **Port**, **Base DN**, and **Search** (since the search scope cannot be set filtered role definitions).

- Select the types of entries to filter from the **For** drop-down list.

The entries can be users, groups, or both.

- Select an attribute from the **Where** drop-down list. The two fields following it refine the search by selecting one of the qualifiers from the drop-down list, such as `contains`, `does not contain`, `is`, or `is not`. Enter an attribute value in the text box. To add additional filters, click **More**. To remove unnecessary filters, click **Fewer**.

- Click **OK**.

6. Click **Test** to try the filter.

A **Filter Test Result** dialog box displays the entries matching the filter.

7. Click **OK**.

The new role appears in the right pane.



NOTE

The `nsRoleDN` attribute is an operational attribute and must be explicitly requested in the search command in the list of search attributes. For example:

```
ldapsearch ... args ... "(uid=scarter)" \* nsRole nsRoleDN
```

The Directory Server Console automatically shows the *nsRoleDN* attribute.

1.2.3. Creating a Nested Role

Nested roles are roles that contain other roles. Before it is possible to create a nested role, another role must exist. When a nested role is created, the Console displays a list of the roles available for nesting. The roles nested within the nested role are specified using the *nsRoleDN* attribute.

To create and add members to a nested role, do the following:

1. Create a new role, as in [Section 1.2.1, “Creating a Managed Role”](#).
2. Click **Members** in the left pane.

A search dialog box appears briefly.
3. In the right pane, select **Nested Role**.
4. Click **Add** to add roles to the list. The members of the nested role are members of other existing roles.

The **Role Selector** dialog box opens.
5. Select a role from the **Available roles** list, and click **OK**.
6. Click **OK** to save the new role.

The new role appears in the right pane.



NOTE

The *nsRoleDN* attribute is an operational attribute and must be explicitly requested in the search command in the list of search attributes. For example:

```
ldapsearch ... args ... "(uid=scarter)" \* nsRole nsRoleDN
```

The Console will automatically show the *nsRoleDN* attribute.

1.2.4. Viewing and Editing an Entry's Roles

To view or edit a role associated with an entry from the Console, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. In the left navigation pane, browse the tree, and select the entry for which to view or edit a role.
3. Select **Set Roles** from the **Object** menu.

The **Roles** dialog box opens.

4. Select the **Managed Roles** tab to display the managed roles to which this entry belongs.

To add a new managed role, click **Add**, and select an available role from the **Role Selector** window. Click **OK**.

To remove a managed role, select it, and click **Remove**.

To edit a managed role associated with an entry, click **Edit**. The **Edit Entry** dialog box opens. Make any changes to the general information or members and click **OK**.

5. Select the **Other Roles** tab to view the filtered or nested roles to which this entry belongs.

Click **Edit** to make changes to any filtered or nested roles associated with the entry. Click **OK**.

6. Click **OK** to save the changes.

1.2.5. Modifying a Role Entry

To edit an existing role, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the navigation tree in the left pane to locate the base DN for the role. Roles are listed in the right pane with other entries.
3. Double-click the role.

The **Edit Entry** dialog box appears.

4. Click **General** in the left pane to change the role name and description.
5. Click **Members** in the left pane to change the members of managed and nested roles or to change the filter of a filtered role.
6. Click **OK** to save the changes.

1.2.6. Making a Role Inactive

Members of a role can be temporarily disabled by inactivating the role to which they belong. Inactivating a role inactivates the entries possessed by the role, not the role itself.

To temporarily disable the members of a role, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the navigation tree in the left pane to locate the base DN for the role. Roles appear in the right pane with other entries.
3. Select the role. Select Inactivate from the **Object** menu.

Alternatively, right-click the role and select Inactivate from the menu.

The role is inactivated.

To see the inactivated entries, select Inactivation State from the **View** menu. A red slash through the role icon indicates that the role has been inactivated.

1.2.7. Reactivating a Role

To reactivate a disabled role:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the navigation tree in the left pane to locate the base DN for the role. Roles appear in the right pane with other entries.
3. Select the role. Select Activate from the **Object** menu.

Alternatively, right-click the role and select Activate from the menu.

The role is reactivated.

To see inactivated entries, select **Inactivation State** from the **View > Display** menu. The role icon appears as normal, indicating that the role is active.

1.2.8. Deleting a Role

Deleting a role deletes the role only, not its members. To delete a role, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the navigation tree in the left pane to locate the base DN for the role. Roles appear in the right pane with other entries.
3. Right-click the role, and select **Delete**.

A dialog box appears to confirm the deletion. Click **Yes**.



NOTE

Deleting a role deletes the role entry but does not delete the `nsRoleDN` attribute for each role member. To delete the `nsRoleDN` attribute for each role member, enable the Referential Integrity plug-in, and configure it to manage the `nsRoleDN` attribute. For more information on the Referential Integrity plug-in, see [Section 5, “Maintaining Referential Integrity”](#).

1.3. Managing Roles Using the Command-Line

Roles inherit from the `ldapsubentry` object class, which is defined in the ITU X.509 standard. In addition, each type of role has two specific object classes that inherit from the `nsRoleDefinition` object class. Once a role is created, members are assigned to it as follows:

- Members of a managed role have the `nsRoleDN` attribute in their entry.
- Members of a filtered role are entries that match the filter specified in the `nsRoleFilter` attribute.
- Members of a nested role are members of the roles specified in the `nsRoleDN` attributes of the nested role definition entry.

[Table 5.1, “Object Classes and Attributes for Roles”](#) lists the object classes and attributes associated with each type of role.

Role Type	Object Classes	Attributes
Managed Role	<code>nsSimpleRoleDefinition</code> <code>nsManagedRoleDefinition</code>	<code>description</code> (optional)
Filtered Role	<code>nsComplexRoleDefinition</code> <code>nsFilteredRoleDefinition</code>	<code>nsRoleFilter</code> <code>Description</code> (optional)
Nested Role	<code>nsComplexRoleDefinition</code> <code>nsNestedRoleDefinition</code>	<code>nsRoleDN</code> <code>Description</code> (optional)

Table 5.1. Object Classes and Attributes for Roles

The attributes `nsRole` and `nsRoleDN` are operational attributes. This means that they are not present in the schema of the entry and may be added to any entry, regardless of schema. This

also means that these attributes must be explicitly requested in the search attributes list in search requests. For example, this `ldapsearch` command lists all of the roles (values of `nsRole`), all of the managed roles (values of `nsRoleDN`), and all of the regular attributes in the entry matched by `uid=scarter`.

```
ldapsearch ... args ... "(uid=scarter)" \* nsRole nsRoleDN
```

Similarly for the role definition entries, they are operational entries and are not returned by default with regular searches. This means that if roles are defined under the `ou=People,dc=example,dc=com` subtree, for example, the following `ldapsearch` command will not return the role definitions for any entry:

```
ldapsearch -s sub -b ou=People,dc=example,dc=com "(objectclass=*)"
```

To see the role definitions entries, use the special search filter `"(objectclass=ldapSubEntry)"` with `ldapsearch`. The special filter can be added to any other search filter, using OR (`|`):

```
ldapsearch -s sub -b ou=People,dc=example,dc=com  
"(|(objectclass=*)(objectclass=ldapSubEntry))"
```

This search shows all regular entries in addition to role definition entries in the `ou=People,dc=example,dc=com` subtree. The Console automatically shows all of the role entries.



NOTE

In some cases, the value of the `nsRoleDNattribute` must be protected with an ACL, as the attribute is writable. For more information about security and roles, see [Section 1.4, "Using Roles Securely"](#).

1.3.1. Examples: Managed Role Definition

Example Corporation's administrator is creating a role to be assigned to all marketing staff by doing the following:

1. Run `ldapmodify`:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

2. Create the managed role entry, containing 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
```

3. Assign the role to a marketing staff member named Bob, using `ldapmodify`:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389

dn: cn=Bob,ou=people,dc=example,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
```

The `nsRoleDN` attribute in the entry indicates that the entry is a member of a managed role, `cn=Marketing,ou=people,dc=example,dc=com`.

1.3.2. Example: Filtered Role Definition

Example Corporation's administrator is creating a filtered role for sales managers.

1. Run `ldapmodify`:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

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
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,ou=people,dc=example,dc=com
objectclass: person
cn: Pat
sn: Pat
userPassword: bigsecret
o: sales managers
```

1.3.3. Example: Nested Role Definition

The Example Corporation administrator is creating a nested role that contains both the marketing staff and sales managers who are members of the roles marketing managed role and the sales filtered role.

1. Run `ldapmodify`:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

2. Create the nested role entry. The nested role has the the `nsNestedRoleDefinition` object class, which inherits from the `LDAPsubentry`, `nsRoleDefinition`, and `nsComplexRoleDefinition` object classes. 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, would be members of this new nested role.

1.4. 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, in some security contexts, it is inappropriate to have such open roles. Consider account inactivation roles. By default, account inactivation roles contain ACIs defined for their suffix. When creating a role, the server administrator decides whether a user can assign themselves to or remove themselves from the role.

For example, user A possesses the managed role, `MR`. The `MR` role has been locked using account inactivation. This means that user A cannot bind to the server because the `nsAccountLock` attribute is computed as `true` for that user. However, suppose the user was already bound and noticed that he is now locked through the `MR` role. If there are no ACIs preventing him, the user can remove the `nsRoleDN` attribute from his entry and unlock himself.

To prevent users from removing the `nsRoleDN` attribute, use the following ACIs depending upon the type of role being used.

- *Managed roles.* For entries that are members of a managed role, use the following ACI to prevent users from unlocking themselves by removing the appropriate `nsRoleDN`:

```
aci: (targetattr="nsRoleDN") (targetattrfilters=
add=nsRoleDN:(!(nsRoleDN=cn=AdministratorRole,
dc=example,dc=com)),
del=nsRoleDN:(!(nsRoleDN=cn=nsManagedDisabledRole,dc=example,dc=com)))
(version3.0;aci allow mod of nsRoleDN by self but not to critical
values; allow(write)
userdn=ldap:///self;)
```

- *Filtered roles.* The attributes that are part of the filter should be protected so that the user cannot relinquish the filtered role by modifying an attribute. The user should not be allowed to add, delete, or modify the attribute used by the filtered role. If the value of the filter attribute is computed, then all attributes that can modify the value of the filter attribute should be protected in the same way.
- *Nested roles.* A nested role is comprised of filtered and managed roles, so the above points should be considered for each of the roles that comprise the nested role.

For more information about account inactivation, see [Section 2, “Inactivating Users and Roles”](#).

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.

- [Section 2.1, “About CoS”](#)
- [Section 2.2, “Managing CoS Using the Console”](#)
- [Section 2.3, “Managing CoS from the Command-Line”](#)
- [Section 2.4, “Creating Role-Based Attributes”](#)
- [Section 2.5, “Access Control and CoS”](#)

2.1. About CoS

Clients of the Directory Server read the attributes on a user's entry. With CoS, some attribute values may not be stored with the entry itself. Instead, they are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of the following two types of entry in 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.

2.1.1. About the CoS Definition Entry

The CoS definition entry is an instance of the `cosSuperDefinition` object class. The CoS definition entry also contains an object class that specifies the type of template entry it uses to generate the entry. There are three different object classes which can be specified, depending upon the type of CoS. The target entries share the same parent as the CoS definition entry.

There are three types of CoS, defined using three types of CoS definition entries:

- *Pointer CoS.* A pointer CoS identifies the template entry using the template DN only.
- *Indirect CoS.* An indirect CoS identifies the template entry using the value of one of the target entry's attributes. For example, an indirect CoS might specify the *manager* attribute of a target entry. The value of the *manager* attribute is then used to identify the template entry.

The target entry's attribute must be single-valued and contain a DN.

- *Classic CoS*. A classic CoS identifies the template entry using a combination of the template entry's base DN and the value of one of the target entry's attributes.

For more information about the object classes and attributes associated with each type of CoS, refer to [Section 2.3, “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.

2.1.2. 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.

2.1.3. How a Pointer CoS Works

An administrator creates a pointer CoS that shares a common postal code with all of the entries stored under `dc=example,dc=com`. The three entries for this CoS appear as illustrated in [Figure 5.1, “Sample Pointer CoS”](#).

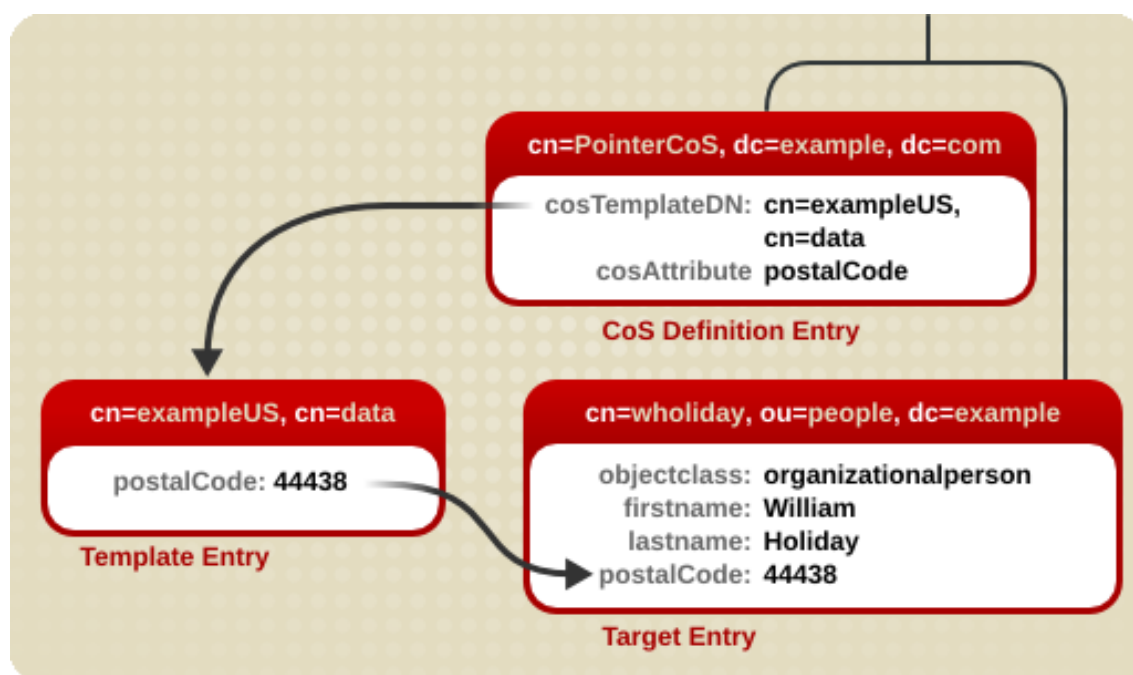


Figure 5.1. Sample Pointer CoS

In this example, the template entry is identified by its DN, `cn=exampleUS, cn=data`, in the CoS definition entry. Each time the `postalCode` attribute is queried on the entry `cn=wholiday, ou=people, dc=example, dc=com`, the Directory Server returns the value available in the template entry `cn=exampleUS, cn=data`.

2.1.4. How an Indirect CoS Works

An administrator creates an indirect CoS that uses the `manager` attribute of the target entry to identify the template entry. The three CoS entries appear as illustrated in [Figure 5.2, “Sample Indirect CoS”](#).

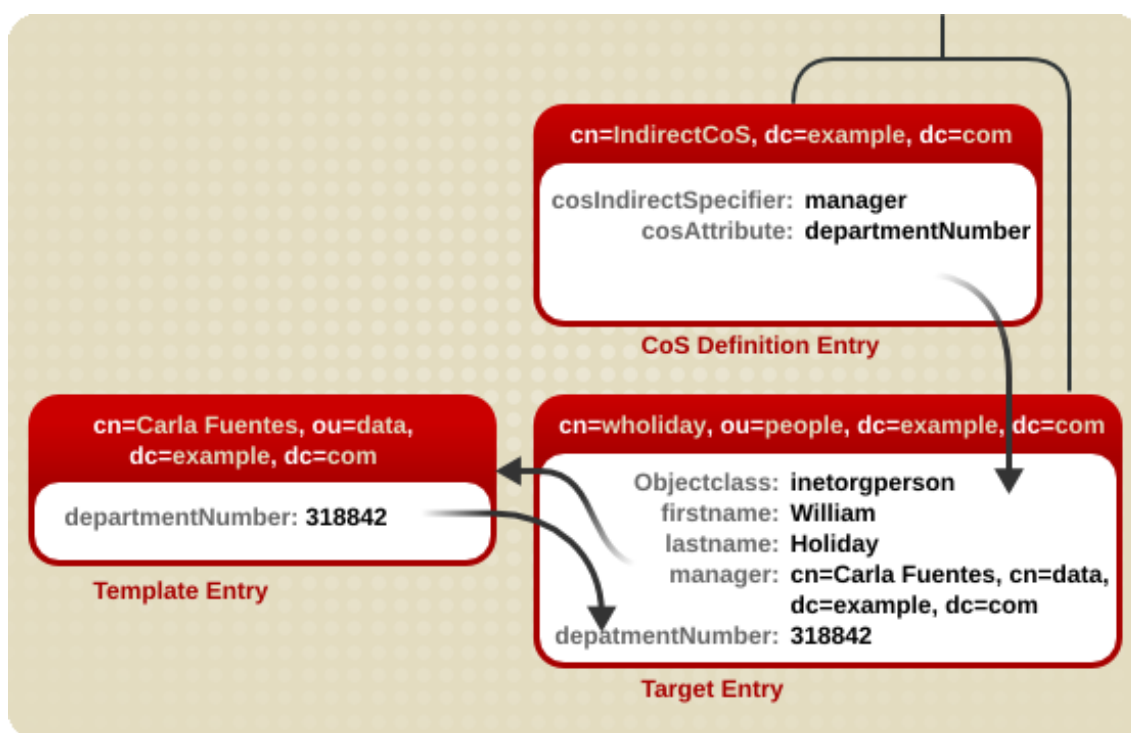


Figure 5.2. Sample Indirect CoS

In this example, the target entry for William Holiday contains the indirect specifier, the *manager* attribute. William's manager is Carla Fuentes, so the *manager* attribute contains a pointer to the DN of the template entry, `cn=Carla Fuentes, ou=people, dc=example, dc=com`. The template entry in turn provides the *departmentNumber* attribute value of 318842.

2.1.5. How a Classic CoS Works

An administrator creates a classic CoS that uses a combination of the template DN and a CoS specifier to identify the template entry containing the postal code. The three CoS entries appear as illustrated in [Figure 5.3, "Sample Classic CoS"](#):

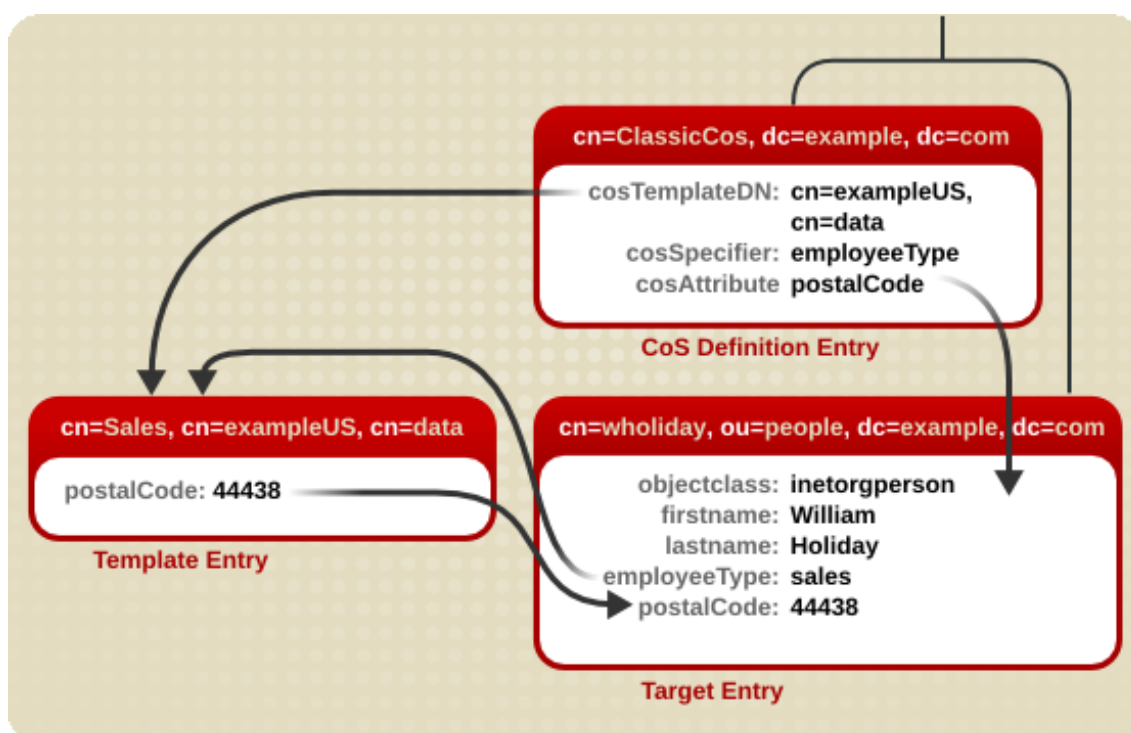


Figure 5.3. Sample Classic CoS

In this example, the CoS definition entry's *cosSpecifier* attribute specifies the *employeeType* attribute. This attribute, in combination with the template DN, identify the template entry as *cn=sales, cn=exampleUS, cn=data*. The template entry then provides the value of the *postalCode* attribute to the target entry.

2.1.6. 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 would be returned, while Ted Morris's would 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 request on CoS-defined attributes, take care when deciding which attributes to generate using a CoS.

2.2. Managing CoS Using the Console

This section describes creating and editing CoS through the Directory Server Console. It includes the following sections:

- [Section 2.2.1, "Creating a New CoS"](#)
- [Section 2.2.2, "Creating the CoS Template Entry"](#)
- [Section 2.2.3, "Editing an Existing CoS"](#)
- [Section 2.2.4, "Deleting a CoS"](#)

2.2.1. Creating a New CoS

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry for the new class of service.
3. Go to the **Object** menu, and select **New > Class of Service**.

Alternatively, right-click the entry and select **New > Class of Service**.

The **Create New Class of Service** dialog opens.

4. Select **General** in the left pane. In the right pane, enter the name of the new class of service in the **Class Name** field. Enter a description of the class in the **Description** field.

5. Click **Attributes** in the left pane. The right pane displays a list of attributes generated on the target entries.

Click **Add** to browse the list of possible attributes and add them to the list.

6. After an attribute is added to the list, a drop-down list appears in the **Class of Service Behavior** column.
 - Select **Does not override target entry attribute** to tell the directory to only return a generated value if there is no corresponding attribute value stored with the entry.
 - Select **Overrides target entry attribute** to make the value of the attribute generated by the CoS override the local value.
 - Select **Overrides target entry attribute and is operational** to make the attribute override the local value and to make the attribute operational, so that it is not visible to client applications unless explicitly requested.
 - Select **Does not override target entry attribute and is operational** to tell the directory to return a generated value only if there is no corresponding attribute value stored with the entry and to make the attribute operational (so that it is not visible to client applications unless explicitly requested).



NOTE

An attribute can only be made operational if it is also defined as operational in the schema. For example, if a CoS generates a value for the *description* attribute, you cannot select **Overrides target entry attribute and is operational** because this attribute is not marked operational in the schema.

7. Click **Template** in the left pane. In the right pane, select how the template entry is identified.
 - *By its DN.* To have the template entry identified by only its DN (a pointer CoS), enter the DN of the template in the **Template DN** field. Click **Browse** to locate the DN on the local server. This will be an exact DN, such as `cn=CoS
template,ou=People,dc=example,dc=com`.
 - *Using the value of one of the target entry's attribute.* To have the template entry identified by the value of one of the target entry's attributes (an indirect CoS), enter the attribute name in the **Attribute Name** field. Click **Change** to select a different attribute from the list of available attributes.
 - *Using both its DN and the value of one of the target entry's attributes.* To have the template entry identified by both its DN and the value of one of the target entry's attributes (a classic CoS), enter both a template DN and an attribute name. The template DN in a classic CoS is more general than for a pointer CoS; it references the suffix or sub suffix where the template entries will be (there can be more than one template).

8. Click **OK**.

2.2.2. Creating the CoS Template Entry

For a pointer CoS or a classic CoS, there must be a template entry, according to the template DN set when the class of service was created. Although the template entries can be placed anywhere in the directory as long as the `cosTemplateDn` attribute reflects that DN, it is best to place the template entries under the CoS itself.

For a pointer CoS, make sure that this entry reflects the exact DN given when the CoS was created. For a classic CoS, the template DN should be recursive, pointing back to the CoS entry itself as the base suffix for the template.

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry that contains the class of service.

The CoS appears in the right pane with other entries.

3. Right-click on the CoS and select **New > Other**.
4. Select `cosTemplate` from the list of object classes.



NOTE

The `LDAPsubentry` object class can be added to a new template entry. Making the CoS template entry an instance of the `LDAPsubentry` object class allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else (for example, if it is a user entry), the `LDAPsubentry` object class does not need to be added to the template entry.

5. The **Property Editor** opens.
6. Select the object classes attribute, and hit **Add Value**. Add the `extensibleObject` object class. This makes it possible to add any attribute available in the directory.
7. Click on the **Add Attribute** button.

Add the `cn` attribute, and give it a value that corresponds to the attribute value in the target entry. For example, if the `manager` attribute is used to set the value for a classic CoS, give the `cn` a value of `uid=bparker,ou=people,dc=example,dc=com`. Alternatively, set it to a role, such as `cn=QA Role,dc=example,dc=com` or a regular attribute value. For example, if the `employeeType` attribute is selected, it can be `full time` or `temporary`.

8. Click the **Add Attribute** button, and add the attributes listed in the CoS. The values used here will be used throughout the directory in the targeted entries.
9. Set the *cospriority*. There may be more than one CoS that applies to a given attribute in an entry; the *cospriority* attribute ranks the importance of that particular CoS. The higher *cospriority* will take precedence in a conflict. The highest priority is 0.

Templates that contain no *cosPriority* attribute are considered the lowest priority. In the case where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily. The behavior for negative *cosPriority* values is not defined in Directory Server; do not enter negative values. Also, the *cosPriority* attribute is not supported by indirect CoS.

10Hit save.

The CoS will be visible in the left navigation pane once there are entries beneath it. For classic CoS, there can be multiple entries, according to the different potential values of the attribute specifier.

2.2.3. Editing an Existing CoS

To edit the description or attributes generated on the target entry of an existing CoS, do the following:

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry that contains the class of service.

The CoS appears in the right pane with other entries.

3. Double-click the CoS.

The **Edit Entry** dialog box appears.

4. Click **General** in the left pane to change the CoS name and description.
5. Click Attributes in the left pane to add or remove attributes generated by the CoS.
6. Click **OK**.

The target entries of the CoS are automatically updated.

2.2.4. Deleting a CoS

1. In the Directory Server Console, select the **Directory** tab.
2. Browse the tree in the left navigation pane, and select the parent entry that contains the class

of service.

The CoS appears in the right pane with other entries.

3. Right-click the CoS, and select **Delete**. A dialog box appears to confirm the deletion. Click **Yes**.

2.3. 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. This section contains the following topics:

- [Section 2.3.1, "Creating the CoS Definition Entry from the Command-Line"](#)
- [Section 2.3.2, "Creating the CoS Template Entry from the Command-Line"](#)
- [Section 2.3.3, "Example of a Pointer CoS"](#)
- [Section 2.3.4, "Example of an Indirect CoS"](#)
- [Section 2.3.5, "Example of a Classic CoS"](#)

2.3.1. 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. [Table 5.2, "CoS Definition Entry Object Classes"](#) lists the object classes associated with each type of CoS definition entry.

CoS Type	Object Classes	Description
Pointer CoS	<code>cosPointerDefinition</code>	Identifies the template entry associated with the CoS definition using the template entry's DN value. The DN of the template entry is specified in the <code>cosTemplateDn</code> attribute.
Indirect CoS	<code>cosIndirectDefinition</code>	Identifies the template entry using the value of one of the target entry's attributes. The attribute of the target entry is specified in the <code>cosIndirectSpecifier</code> attribute.
Classic CoS	<code>cosClassicDefinition</code>	Identifies the template entry

CoS Type	Object Classes	Description
		using both the template entry's DN (as specified in the <i>cosTemplateDn</i> attribute) and the value of one of the target entry's attributes (as specified in the <i>cosSpecifier</i> attribute).

Table 5.2. CoS Definition Entry Object Classes

[Table 5.3, “CoS Definition Entry Attributes”](#) lists attributes that available to use in the CoS definition entries.

Attribute	Definition
<i>cosAttribute</i>	Provides the name of the attribute for which to generate a value. There can be more than one <i>cosAttribute</i> value. This attribute is used by all types of CoS definition entries.
<i>cosIndirectSpecifier</i>	Specifies the attribute value used by an indirect CoS to identify the template entry.
<i>cosSpecifier</i>	Specifies the attribute value used by a classic CoS, which, along with the template entry's DN, identifies the template entry.
<i>cosTemplateDn</i>	Provides the DN of the template entry associated with the CoS definition. Used for pointer CoS and classic CoS only.

Table 5.3. CoS Definition Entry Attributes

The *cosAttribute* attribute allows an additional qualifier after the attribute value. There are four possible qualifiers:

- *Default*. This qualifier indicates that the server only returns a generated value if there is no corresponding attribute value stored with the entry.
- *Override*. This qualifier indicates that the server always returns the value generated by the CoS, even when there is a value stored with the entry.
- *Operational*. This qualifier indicates that the attribute will only be returned if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order

to be returned. When `operational` is used as a qualifier, it works as if `override` and `operational` were specified.



NOTE

An attribute can only be made operational if it is also 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*. This qualifier indicates that the server only returns a generated value if there is no corresponding attribute value stored with the entry and if it is explicitly requested in the search.

If no qualifier is set, `default` is assumed.

For example, a pointer CoS definition entry that contains an `override` qualifier is created as follows:

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode override
```

This pointer CoS definition entry indicates that it is associated with a template entry, `cn=exampleUS,ou=data,dc=example,dc=com`, that generates the value of the *postalCode* attribute. The `override` qualifier indicates that this value will take precedence over the value stored by the entries for the *postalCode* attribute.



NOTE

If an entry contains an attribute value generated by a CoS, the value of the attribute *cannot* be manually updated if it is defined with the `operational` or `override` qualifiers.

For more information about the attributes, refer to the *Directory Server Configuration, Command, and File Reference*.

Table 5.4, “CoS Definitions” describes the CoS definition for each type of CoS.

CoS Type	CoS definition
Pointer CoS	<pre> objectclass: top objectclass: cosSuperDefinition objectclass: cosPointerDefinition cosTemplateDn:DN_string cosAttribute:list_of_attributes qualifier </pre>
Indirect CoS	<pre> objectclass: top objectclass: cosSuperDefinition objectclass: cosIndirectDefinition cosIndirectSpecifier:attribute_name cosAttribute:list_of_attributes qualifier </pre>
Classic CoS	<pre> objectclass: top objectclass: cosSuperDefinition objectclass: cosClassicDefinition cosTemplateDn:DN_string cosSpecifier:attribute_name cosAttribute:list_of_attributes qualifier </pre>

Table 5.4. CoS Definitions

CoS definition entries are *operational* entries and are not returned by default with regular searches. This means that if a CoS is defined under `ou=People,dc=example,dc=com`, for example, the following `ldapsearch` command will not return them:

```
ldapsearch -s sub -b ou=People,dc=example,dc=com "(objectclass=*)"
```

To return the CoS definition entries, add the `ldapSubEntry` object class to the CoS definition entries. For example:

```

dn: cn=pointerCoS,ou=People,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
objectclass: ldapSubEntry
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode override

```

Then use a special search filter, `(objectclass=ldapSubEntry)`, with the search. This filter can

be added to any other search filter using OR (|):

```
ldapsearch -s sub -b ou=People,dc=example,dc=com
"(|(objectclass=*)(objectclass=ldapSubEntry))"
```

This search returns all regular entries in addition to CoS definition entries in the `ou=People,dc=example,dc=com` subtree.



NOTE

The Console automatically shows CoS entries.

2.3.2. 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
```

It is possible to create CoS templates that compete with each other to provide an attribute value. For example, there can be a multi-valued `cosSpecifier` attribute in the CoS definition entry. Specifying the template priority on each template entry determines which template provides the attribute value. Set the template priority using the `cosPriority` attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

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. The behavior for negative *cosPriority* values is not defined in Directory Server; do not enter negative values. Also, the *cosPriority* attribute is not supported by indirect CoS.

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

- [Section 2.3.3, “Example of a Pointer CoS”](#)
- [Section 2.3.4, “Example of an Indirect CoS”](#)
- [Section 2.3.5, “Example of a Classic CoS”](#)

2.3.3. Example of a Pointer CoS

Example Corporation's administrator is creating a pointer CoS that shares a common postal code with all entries in the *dc=example,dc=com* tree.

1. Add a new pointer CoS definition entry to the *dc=example,dc=com* suffix using `ldapmodify`:

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

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

2. Next, add the pointer CoS definition to the *dc=example,dc=com* root suffix.

```
dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
```

```
cosTemplateDn: cn=exampleUS,ou=data,dc=example,dc=com
cosAttribute: postalCode
```

3. Create the template entry.

```
dn: cn=exampleUS,ou=data,dc=example,dc=com
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.

2.3.4. 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` as follows:

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

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

2. Add the indirect CoS definition to the `dc=example,dc=com` root suffix as follows:

```
dn: cn=indirectCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

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.

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

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

2. Add the indirect CoS definition to the *dc=example,dc=com* root suffix.

```
dn: cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=classicCoS,dc=example,dc=com
cosSpecifier: businessCategory
cosAttribute: postalCode override
```

3. Create the template entries for the sales and marketing departments. Add the CoS attributes to the template entry. The *cn* of the template sets the value of the *businessCategory* attribute in the target entry, and then the attributes are added or overwritten according to the value in the template:

```
dn: cn=sales,cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 44438

dn: cn=marketing,cn=classicCoS,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
postalCode: 99111
```

The classic CoS definition entry applies to all entries under the *dc=example,dc=com* suffix. Depending upon the combination of the *businessCategory* attribute found in the entry and the *cosTemplate* DN, it can arrive at one of two templates. One, the sales template, provides a postal code specific to employees in the sales department. The marketing template provides a postal code specific to employees in the marketing department.

2.4. 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: o=managers
Description: filtered role for managers
```

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.

2.5. 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.

3. 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.

Views create a directory tree similar to the regular hierarchy, such as using organizational unit entries for subtrees, but views entries have an additional object class (`nsview`) and a filter attribute (`nsviewfilter`) that set up a filter for the entries which belong in that view. Once the view container entry is added, all of the entries that match the view filter instantly populate the view. The target entries only *appear* to exist in the view; their true location never changes. For example, a view may be created as `ou=Location Views`, and a filter is set for `l=Mountain View`. Every entry, such as `cn=Jane Smith,l=Mountain View,ou=People,dc=example,dc=com`, is immediately listed under the `ou=Location Views` entry, but the real `cn=Jane Smith` entry remains in the `ou=People,dc=example,dc=com` subtree.

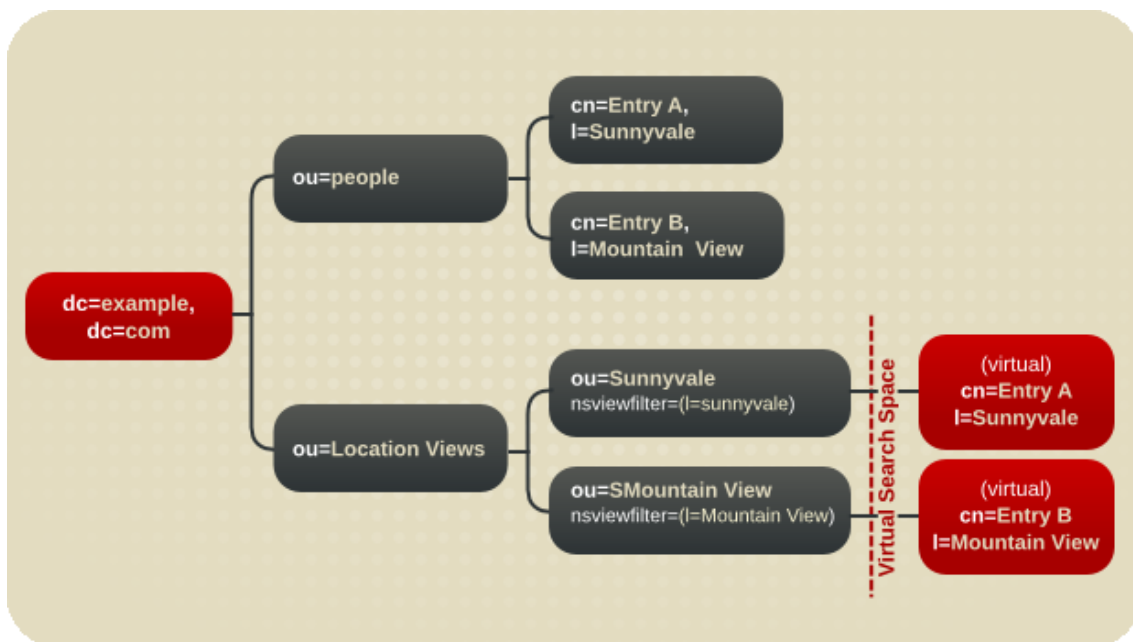


Figure 5.4. A Directory Tree with a Virtual DIT View hierarchy

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.



TIP

There is a sample LDIF file with example views entries, `Example-views.ldif`, installed with Directory Server. This file is in the `/usr/share/dirsrv/data` directory on Red Hat Enterprise Linux and Solaris and the `/opt/dirsrv/share/data` directory on HP-UX.

3.1. Creating Views in the Console

To create a view in the Directory Server Console, do the following:

1. Select the **Directory** tab.
2. In the left navigation tree, create a suffix to hold the views. For instance, for views based on the locality (1) attribute, name this organizational unit `Location Views`.
3. Right-click `ou=Location Views`, and select **New > Other**.
4. Select `nsview` from the **New Object** menu, and hit **OK**.

5. In the **Property Editor** window, hit the **Add Value** button, and add the organization unit object class.
6. Name the organization unit according to how to organize the views. For instance, `ou=Sunnyvale`. Make the `ou` attribute the naming attribute.
7. Hit the **Add Attribute** button, and add the `nsviewfilter` attribute.
8. Create a filter that reflects the views. For example:

```
(l=Sunnyvale)
```

9. Hit **OK** to close the attributes box, and hit **OK** again to save the new view entry.

The new view is immediately populated with any entries matching the search filter, and any new entries added to directory are automatically included in the view.

3.2. Deleting Views from the Directory Server Console

To delete a view from the Directory Server Console, do the following:

1. Select the **Directory** tab.
2. Select the view to delete, such as `ou=Sunnyvale,ou=LocationViews,dc=example,dc=com`. To delete all the views, delete the entire sub suffix, `ou=LocationViews,dc=example,dc=com`.
3. Right-click the entry, and select **Delete** from the drop-down menu.

Alternatively, highlight the entry, and select the **Object** menu and then select **Delete**.

4. A dialog box appears to confirm the deletion of the entry. Click **Yes**.

3.3. Creating Views from the Command Line

To create a view from the command line, do the following:

1. Use the `ldapmodify` utility to bind to the server and prepare it to add the new view entry to the configuration file.

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

2. Add the new views container entry, in this example, under the `dc=example,dc=com` root suffix. This entry must have the `nsview` object class and the `nsViewFilter` attribute. The `nsViewFilter` attribute sets the attribute-value which identifies entries that belong in the

view.

```
dn: ou=Example View,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
objectClass: nsview
ou=Example View
nsViewFilter: l=Mountain View
description: Example View
```

3.4. Deleting Views from the Command Line

To delete a view from the command line, do the following:

1. Use the `ldapdelete` utility to bind to the server and prepare it to remove the view entry to the configuration file.

```
ldapdelete -D "cn=directory manager" -w secret -h host -p 389 "ou=Example
View,dc=example,dc=com"
```

2. Remove the view entry. It is not necessary to remove any entries included in the view.

```
dn: ou=Example View,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
objectClass: nsview
ou=Example View
nsViewFilter: l=Mountain View
description: Example View
```

4. Using Groups

Groups are a mechanism for associating entries for ease of administration. This mechanism was provided with previous versions of Directory Server and should be used primarily for compatibility with older versions of the server.

4.1. Managing Static Groups

Static groups organize entries by specifying the same group value in the DN attribute of any number of users. This section includes the following procedures for creating and modifying static groups:

- [Section 4.1.1, “Adding a New Static Group”](#)

- [Section 4.1.2, “Modifying a Static Group”](#)



NOTE

If a user has an entry on a remote Directory Server (for example, in a chained database), different from the Directory Server which has the entry that defines the static group, then use the Referential Integrity plug-in to ensure that deleted user entries are automatically deleted from the static group, but there are some performance and access control considerations. For more information about using referential integrity with chaining, refer to [Section 3.1, “Configuring the Chaining Policy”](#).

4.1.1. Adding a New Static Group

1. In the Directory Server Console, select the **Directory** tab.
2. In the left pane, right-click the entry under which to add a new group, and select **New > Group**.

Alternatively, go to the **Object** menu and select **New > Group**.

3. Click **General** in the left pane. Type a name for the new group in the **Group Name** field.

The group name is required.

4. Enter a description of the new group in the **Description** field.
5. Click **Members** in the left pane. In the right pane, select the **Static Group** tab. Click **Add** to add new members to the group.

The standard **Search users and groups** dialog box appears.

6. In the **Search** drop-down list, select what sort of entries to search for (users, groups, or both) then click **Search**. Select one of the entries returned, and click **OK**.
7. Click **Languages** in the left pane to add language-specific information for the group.
8. Click **OK** to create the new group. It appears in the right pane.

4.1.2. Modifying a Static Group

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

The directory contents appear in the left pane.

2. Double-click the entry to modify, or select **Open** from the **Object** menu.

The **Edit Group** dialog box appears.

3. Click **OK**. To view the changes, go to the **View** menu, and select **Refresh**.



NOTE

The Console for managing static groups may not display all possible selections during a search operation if there is no VLV index for users' search. This problem occurs only when the number of users is 1000 or more and there is no VLV index for search. To work around the problem, create a VLV index for the users suffix with the filter `(objectclass=person)` and scope `sub-tree`.

4.2. Managing Dynamic Groups

Dynamic groups filter users based on their DN and include them in a single group. This section contains the following procedures for creating and modifying dynamic groups:

- [Section 4.2.1, “Adding a New Dynamic Group”](#)
- [Section 4.2.2, “Modifying a Dynamic Group”](#)

4.2.1. Adding a New Dynamic Group

1. Follow the steps of [Section 4.1.1, “Adding a New Static Group”](#).
2. Click **Members** in the left pane. In the right pane, select the **Dynamic Group** tab. Click **Add** to create a LDAP URL for querying the database.

The standard **Construct and Test LDAP URL** dialog box opens.

3. Enter an LDAP URL in the text field or select **Construct** to be guided through the construction of an LDAP URL.
4. Click **Languages** in the left pane to add language-specific information for the group.
5. Click **OK**. The new group appears in the right pane.

4.2.2. Modifying a Dynamic Group

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

The directory contents appear in the left pane.

2. Double-click the entry to modify, or select **Properties** from the **Object** menu.

The **Edit Group** dialog box appears.

3. Make any changes to the group information. Click **OK**.

To view the changes, go to the **View** menu, and select **Refresh**.



NOTE

The Console for managing dynamic groups may not display all possible selections during a search operation if there is no VLV index for users' search. This problem can occur when the number of users is 1000 or more and there is no VLV index for search. To work around the problem, create a VLV index for the users suffix with the filter `(objectclass=person)` and scope `sub-tree`.

Managing Access Control

Red Hat Directory Server allows you to control access to your directory. This chapter describes the how to implement *access control*. To take full advantage of the power and flexibility of access control, while you are in the planning phase for your directory deployment, define an access control strategy as an integral part of your overall security policy.

1. Access Control Principles

The mechanism which defines user access is called *access control*. When the server receives a request, it uses the authentication information provided by the user in the bind operation and the access control instructions (ACIs) defined in the server to allow or deny access to directory information. The server can allow or deny permissions for actions on entries like read, write, search, and compare. The permission level granted to a user may depend on the authentication information provided.

Access control in Directory Server is flexible enough to provide very precise rules on when the ACIs are applicable:

- For the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes.
- For a specific user, all users belonging to a specific group or role, or all users of the directory.
- For a specific location such as an IP address or a DNS name.

1.1. ACI Structure

Access control instructions are stored in the directory as attributes of entries. The `aci` attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by the Directory Server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The `aci` attribute is returned in an `ldapssearch` operation if specifically requested.

The three main parts of an ACI statement are:

- Target
- Permission
- Bind Rule

The permission and bind rule portions of the ACI are set as a pair, also called an *access control rule* (ACR). The specified permission is granted or denied depending on whether the accompanying rule is evaluated to be true.

1.2. ACI Placement

If an entry containing an ACI does not have any child entries, the ACI applies to that entry only. If the entry has child entries, the ACI applies to the entry itself and all entries below it. As a direct consequence, when the server evaluates access permissions to any given entry, it verifies the ACIs for every entry between the one requested and the directory suffix, as well as the ACIs on the entry itself.

The *aci* attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

An ACI created on an entry can be set so it does not apply directly to that entry but to some or all of the entries in the subtree below it. The advantage of this is that general ACIs can be placed at a high level in the directory tree that effectively apply to entries more likely to be 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.

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 as close as possible to leaf entries.



NOTE

ACIs placed in the root DSE entry apply only to that entry.

1.3. ACI Evaluation

To evaluate the access rights to a particular entry, the server compiles a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored on the Directory Server. ACIs are evaluated across all of the databases for a particular Directory Server but not across all Directory Server instances.

The evaluation of this list of ACIs is done based on the semantics of the ACIs, not on their placement in the directory tree. This means that ACIs that are close to the root of the directory tree do not take precedence over ACIs that are closer to the leaves of the directory tree.

For Directory Server ACIs, the *precedence rule* is that ACIs that deny access take precedence over ACIs that allow access. Between ACIs that allow access, union semantics apply, so there is no precedence.

For example, if you deny write permission at the directory's root level, then none of the users can write to the directory, regardless of the specific permissions you grant them. To grant a specific user write permissions to the directory, you have to restrict the scope of the original denial for write permission so that it does not include the user.

1.4. ACI Limitations

When creating an access control policy for your directory service, you need to be aware of the following restrictions:

- If your directory tree is distributed over several servers using the chaining feature, some restrictions apply to the keywords you can use in access control statements:
 - ACIs that depend on group entries (`groupdn` keyword) must be located on the same server as the group entry. If the group is dynamic, then all members of the group must have an entry on the server, too. If the group is static, the members' entries can be located on remote servers.
 - ACIs that depend on role definitions (`roledn` keyword) must be located on the same server as the role definition entry. Every entry that is intended to have the role must also be located on the same server.

However, you can match values stored in the target entry with values stored in the entry of the bind user; for example, using the `userattr` keyword. Access is evaluated normally even if the bind user does not have an entry on the server that holds the ACI.

For more information on how to chain access control evaluation, see [Section 3.5, “Database Links and Access Control Evaluation”](#).

- Attributes generated by class of service (CoS) cannot be used in all ACI keywords. Specifically, you should not use attributes generated by CoS with the following keywords:
 - `targetfilter` ([Section 3.2.4, “Targeting Entries or Attributes Using LDAP Filters”](#))
 - `targetfilters` ([Section 3.2.2, “Targeting Attributes”](#))
 - `userattr` ([Section 4.5.1, “Using the userattr Keyword”](#))

If you create target filters or bind rules that depend on the value of attributes generated by CoS, the access control rule will not work. For more information on CoS, see [Chapter 5, Managing Entries with Roles, Class of Service, and Views](#).

- Access control rules are always evaluated on the local server. Therefore, it is not necessary to specify the hostname or port number of the server in LDAP URLs used in ACI keywords. If you do, the LDAP URL is not taken into account at all. For more information on LDAP URLs, see [Appendix C, LDAP URLs](#).

2. Default ACIs

When the Administration Server is set up, the following default ACIs apply to the directory information stored in the `userRoot` database:

- Users can modify a list of common attributes in their own entries, including the `mail`, `telephoneNumber`, `userPassword`, and `seeAlso` attributes. Operational and most of the

security attributes, such as `aci`, `nsroledn`, and `passwordExpirationTime`, cannot be modified by users.

- Users have anonymous access to the directory for search, compare, and read operations.
- The administrator (by default `uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot`) has all rights except proxy rights.
- All members of the `Configuration Administrators` group have all rights except proxy rights.
- All members of the `Directory Administrators` group have all rights except proxy rights.
- `Server Instance Entry (SIE)` group.

The `NetscapeRoot` subtree has its own set of default ACIs:

- All members of the `Configuration Administrators` group have all rights on the `NetscapeRoot` subtree except proxy rights.
- Users have anonymous access to the `NetscapeRoot` subtree for search and read operations.
- All authenticated users have search, compare, and read rights to configuration attributes that identify the Administration Server.
- Group expansion.

The following sections explain how to modify these default settings.

3. Creating ACIs Manually

You can create access control instructions manually using LDIF statements and add them to your directory tree using the `ldapmodify` utility, similar to the instructions in [Section 4, “LDIF Update Statements”](#). The following sections explain in detail how to create the LDIF statements.



TIP

LDIF ACI statements can be very complex. However, if you are setting access control for a large number of directory entries, using LDIF is the preferred because it is faster than using the Console. To familiarize yourself with LDIF ACI statements, however, you may want to use the Directory Server Console to set the ACI and then click the **Edit Manually** button on the **Access Control Editor**. This shows you the correct LDIF syntax. If your operating system allows it, you can even copy the LDIF from the **Access Control Editor** and paste it into your LDIF file.

3.1. The ACI Syntax

The *aci* attribute uses the following syntax:

```
aci: (target)(version 3.0;acl "name";permissionbind_rules;)
```

- *target* specifies the entry, attributes, or set of entries and attributes for which to control access. The target can be a distinguished name, one or more attributes, or a single LDAP filter. The target is an optional part of the ACI.
- *version 3.0* is a required string that identifies the ACI version.
- *name* is a name for the ACI. The name can be any string that identifies the ACI. The ACI name is required.
- *permission* specifically outlines what rights are being allowed or denied; for example, read or search rights.
- *bind_rules* specify the credentials and bind parameters that a user has to provide to be granted access. Bind rules can also specifically deny access to certain users or groups of users.

You can have multiple permission-bind rule pairs for each target. This allows you to set multiple access controls for a given target efficiently. For example:

```
target(permissionbind_rule)(permissionbind_rule)...
```

If you have several ACRs in one ACI statement, the syntax is in the following form:

```
aci: (target)(version 3.0;acl "name";permissionbind_rule;  
permissionbind_rule; ... permissionbind_rule;)
```

The following is an example of a complete LDIF ACI:

```
aci: (target="ldap:///uid=bjensen,dc=example,dc=com")(targetattr=*)  
(version 3.0;acl "aci1";allow (write) userdn="ldap:///self";)
```

In this example, the ACI states that the user *bjensen* has rights to modify all attributes in her own directory entry.

3.2. Defining Targets

The target identifies to what the ACI applies. If the target is not specified, the ACI applies to the entry containing the *aci* attribute and to the entries below it. A target can be any of the following:

- A directory entry or all of the entries in a subtree, as described in [Section 3.2.1, “Targeting a Directory Entry”](#).
- Attributes of an entry, as described in [Section 3.2.2, “Targeting Attributes”](#).
- A set of entries or attributes that match a specified LDAP filter, as described in [Section 3.2.4, “Targeting Entries or Attributes Using LDAP Filters”](#).
- An attribute value, or a combination of values, that match a specified LDAP filter, as described in [Section 3.2.5, “Targeting Attribute Values Using LDAP Filters”](#).

The general syntax for a target is as follows:

```
(keyword = "expression")  
(keyword != "expression")
```

- *keyword* indicates the type of target.
- equal (=) indicates that the target is the object specified in the *expression*, and not equal (!=) indicates the target is not the object specified in the *expression*.
- *expression* identifies the target.

The quotation marks (" ") around *expression* are required. What you use for *expression* is dependent upon the *keyword* that you supply.

[Table 6.1, “LDIF Target Keywords”](#) lists each keyword and the associated expressions.

Keyword	Valid Expressions	Wildcard Allowed
target	<i>ldap:///distinguished_name</i>	Yes
targetattr	<i>attribute</i>	Yes
targetfilter	<i>LDAP_filter</i>	Yes
targetattrfilters	<i>LDAP_operation:LDAP_filter</i>	Yes

Table 6.1. LDIF Target Keywords

In all cases, you must keep in mind that when you place an ACI on an entry, if it is not a leaf entry, the ACI also applies to all entries below it. For example, if you target the entry *ou=accounting,dc=example,dc=com*, the permissions you set apply to all entries in the

accounting branch of the `example.com` tree.

As a counter example, if you place an ACL on the `ou=accounting,dc=example,dc=com` entry, you cannot target the `uid=sarette,ou=people,dc=example,dc=com` entry because it is not located under the accounting tree.

Be wary of using `!=` when specifying an attribute to deny. ACLs are treated as a logical OR, which means that if you created two ACLs as shown below, the result allows all values of the target attribute.

```
acl1: ( target=...)( targetattr!=a )(version 3.0; acl "name";allow (...)..
acl2: ( target=...)( targetattr!=b )(version 3.0; acl "name";allow (...)..
```

The first ACL (`acl1`) allows `b` and the second ACL (`acl2`) allows `a`. The result of these two ACLs is the same as the one resulting from using an ACL of the following form:

```
acl3: ( targetattr="*" ) allow (...) ...
```

In the second example, nothing is denied, which could give rise to security problems.

When you want to deny access to a particular attribute, use `deny` in the permissions clause rather than using `allow` with `(targetattr != value)`. For example, usages such as these are recommended:

```
acl1: ( target=...)( targetattr=a )(version 3.0; acl "name";deny (...)..
acl2: ( target=...)( targetattr=b )(version 3.0; acl "name";deny (...)..
```

3.2.1. Targeting a Directory Entry

To target a directory entry (and the entries below it), you must use the `target` keyword. The `target` keyword can accept a value of the following format:

```
target="ldap:///distinguished_name"
```

This identifies the distinguished name of the entry to which the access control rule applies. For example:

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



NOTE

If the DN of the entry to which the access control rule applies contains a comma, escape the comma with a single backslash (`\`), such as

```
(target="ldap:///uid=lfuentes,dc=example.com Bolivia\,S.A.").
```

Wildcards can be used when targeting a distinguished name using the `target` keyword. The wildcard indicates that any character or string or substring is a match for the wildcard. Pattern matching is based on any other strings that have been specified with the wildcard.

The following are legal examples of wildcard usage:

- `(target="ldap:///uid=*,dc=example,dc=com")` — Matches every entry in the entire `example.com` tree that has the `uid` attribute in the entry's RDN.
- `(target="ldap:///uid=*Anderson,dc=example,dc=com")` — Matches every entry directly under the `example.com` node with a `uid` ending in `Anderson`.
- `(target="ldap:///uid=C*A,dc=example,dc=com")` — Matches every entry directly under the `example.com` node with a `uid` beginning with `C` and ending with `A`.
- `(target="ldap:///uid=*,dc=example,dc=com")` — Matches every entry in the entire `example.com` tree that has the `uid` attribute in the entry's RDN.
- `(target="ldap:///uid=*,ou=*,dc=example,dc=com")` — Matches every entry in the `example.com` tree whose distinguished name contains the `uid` and `ou` attributes. Thus, `uid=fchen,ou=Engineering,dc=example,dc=com` OR `uid=claire,ou=Engineering,ou=people,dc=example,dc=com` would match, but `uid=bjensen,dc=example,dc=com ou=Engineering,dc=example,dc=com` would not.

Depending on the position of the wildcard, it can apply to the full DN, not only to attribute values. Therefore, the wildcard can be used as a substitute for portions of the DN. For example, `uid=andy*,dc=example,dc=com` targets all the directory entries in the entire `example.com` tree with a matching `uid` attribute and not just the entries that are immediately below the `dc=example,dc=com` node. In other words, this target matches with longer expressions such as `uid=andy,ou=eng,dc=example,dc=com` OR `uid=andy,ou=marketing,dc=example,dc=com`.



NOTE

You cannot use wildcards in the suffix part of a distinguished name. That is, if your directory uses the suffixes `c=US` and `c=GB`, then you cannot use `(target="ldap:///dc=example,c=*")` as a target to reference both suffixes. Neither can you use a target such as `uid=bjensen,dc=*.com`.

3.2.2. Targeting Attributes

In addition to targeting directory entries, you can also target one or more attributes included in

the targeted entries. This is useful to deny or allow access to partial information about an entry. For example, you could allow access to only the common name, surname, and telephone number attributes of a given entry while denying access to sensitive information such as passwords.

You can specify that the target is equal or is not equal to a specific attribute. The attributes you supply do not need to be defined in the schema. This absence of schema checking makes it possible to implement an access control policy when you set up your directory service for the first time, even if the ACLs you create do not apply to the current directory content.

To target attributes, use the `targetattr` keyword. The keyword uses the following syntax:

```
(targetattr = "attribute")
```

You can target multiple attributes by using the `targetattr` keyword with the following syntax:

```
(targetattr = "attribute1 || attribute2 ... || attributen")
```

attributeX is the name of the targeted attribute. For example, this targets the common name (*cn*) attribute:

```
(targetattr = "cn")
```

To target an entry's common name, surname, and UID attributes, use the following:

```
(targetattr = "cn || sn || uid")
```

The attributes specified in the `targetattr` keyword apply to the entry that the ACI is targeting and to all the entries below it. If you target the password attribute on the entry `uid=bjensen,ou=Marketing,dc=example,dc=com`, only the password attribute on the `bjensen` entry is affected by the ACI because it is a leaf entry.

If, however, you target the tree's branch point `ou=Marketing,dc=example,dc=com`, then all the entries beneath the branch point that can contain a password attribute are affected by the ACI.

3.2.3. Targeting Both an Entry and Attributes

By default, the entry targeted by an ACI containing a `targetattr` keyword is the entry on which the ACI is placed. That is, putting an ACI such as `aci: (targetattr = "uid") (access_control_rules;)` on the `ou=Marketing,dc=example,dc=com` entry means that the ACI applies to the entire `Marketing` subtree. However, you can also explicitly specify a target using the `target` keyword:

```
aci:  
(target="ldap:///ou=Marketing,dc=example,dc=com")(targetattr="uid")(access_control_rules;)
```

The order in which you specify the `target` and the `targetattr` keywords is not important.

3.2.4. Targeting Entries or Attributes Using LDAP Filters

You can use LDAP filters to target a group of entries that match certain criteria. To do this, you must use the `targetfilter` keyword with an LDAP filter. The syntax of the `targetfilter` keyword is as follows:

```
(targetfilter = "LDAP_filter")
```

LDAP_filter is a standard LDAP search filter. For more information on the syntax of LDAP search filters, see [Appendix B, Finding Directory Entries](#).

For example, suppose that all entries in the accounting department include the attribute-value pair `ou=accounting`, and all entries in the engineering department include the attribute-value pair `ou=engineering` subtree. The following filter targets all the entries in the accounting and engineering branches of the directory tree:

```
(targetfilter = "(|(ou=accounting)(ou=engineering))")
```

This type of filter targets whole entries. You can associate the `targetfilter` and the `targetattr` keywords to create ACIs that apply to a subset of attributes in the targeted entries.

The following LDIF example allows members of the Engineering Admins group to modify the `departmentNumber` and `manager` attributes of all entries in the `Engineering` business category. This example uses LDAP filtering to select all entries with `businessCategory` attributes set to `Engineering`:

```
dn: dc=example,dc=com
objectClass: top
objectClass: organization
aci: (targetattr="departmentNumber || manager")
    (targetfilter="(businessCategory=Engineering)")
    (version 3.0; acl "eng-admins-write"; allow (write)
    groupdn = "ldap:///cn=Engineering Admins, dc=example,dc=com";)
```



TIP

Although using LDAP filters can be useful when you are targeting entries and attributes that are spread across the directory, the results are sometimes unpredictable because filters do not directly name the object for which you are managing access. The set of entries targeted by a filtered ACI is likely to change as attributes are added or deleted. Therefore, if you use LDAP filters in ACIs, you

should verify that they target the correct entries and attributes by using the same filter in an `ldapsearch` operation.

3.2.5. Targeting Attribute Values Using LDAP Filters

You can use access control to target specific attribute values. This means that you can grant or deny permissions on an attribute if that attribute's value meets the criteria defined in the ACI. An ACI that grants or denies access based on an attribute's value is called a value-based ACI.

For example, you might grant all users in your organization permission to modify the `nsroledn` attribute in their own entry. However, you would also want to ensure that they do not give themselves certain key roles, such as `Top Level Administrator`. LDAP filters are used to check that the conditions on attribute values are satisfied.

To create a value-based ACI, you must use the `targattrfilters` keyword with the following syntax:

```
(targattrfilters="add=attr1:F1 && attr2:F2... && attrn:Fn,del=attr1:F1 &&
attr2:F2 ... && attrn:Fn")
```

- `add` represents the operation of creating an attribute.
- `del` represents the operation of deleting an attribute.
- `attrx` represents the target attributes.
- `Fx` represents filters that apply only to the associated attribute.

When creating an entry, if a filter applies to an attribute in the new entry, then each instance of that attribute must satisfy the filter. When deleting an entry, if a filter applies to an attribute in the entry, then each instance of that attribute must also satisfy the filter.

When modifying an entry, if the operation adds an attribute, then the add filter that applies to that attribute must be satisfied; if the operation deletes an attribute, then the delete filter that applies to that attribute must be satisfied. If individual values of an attribute already present in the entry are replaced, then both the add and delete filters must be satisfied.

For example, consider the following attribute filter:

```
(targattrfilters="add=nsroledn:(!(nsroledn=cn=superAdmin)) &&
telephoneNumber:(telephoneNumber=123*)")
```

This filter can be used to allow users to add any role (`nsroledn` attribute) to their own entry,

except the `superAdmin` role. It also allows users to add a telephone number with a 123 prefix.



NOTE

You cannot create value-based ACIs from the Directory Server Console.

3.2.6. Targeting a Single Directory Entry

Targeting a single directory entry is not straightforward because it goes against the design philosophy of the access control mechanism. However, it can be done in either of two ways:

- By creating a bind rule that matches user input in the bind request with an attribute value stored in the targeted entry. For more details, see [Section 4.5, “Defining Access Based on Value Matching”](#).
- By using the `targetattr` and `targetfilter` keywords.

You can use the `targetattr` keyword to specify an attribute that is only present in the entry you want to target, and not in any of the entries below your target. For example, if you want to target `ou=people,dc=example,dc=com`, and there are not any organizational units (`ou`) defined below that node, you could specify an ACI that contains `targetattr=ou`.

A safer method is to use the `targetfilter` keyword and to specify explicitly an attribute value that appears in the entry alone. For example, during the installation of the Directory Server, the following ACI is created:

```
aci: (targetattr="*)(targetfilter=(o=NetscapeRoot))(version 3.0;  
    acl "Default anonymous access"; allow (read, search)  
    userdn="ldap:///anyone";)
```

This ACI can apply only to the `o=NetscapeRoot` entry.

The risk associated with these method is that your directory tree might change in the future, and you would have to remember to modify this ACI.

3.3. Defining Permissions

Permissions specify the type of access you are allowing or denying. You can either allow or deny permission to perform specific operations in the directory. The various operations that can be assigned are known as *rights*.

There are two parts to setting permissions:

- Allowing or denying access

- Assigning rights

3.3.1. Allowing or Denying Access

You can either explicitly allow or deny access permissions to the directory tree.



NOTE

From the Directory Server Console, you cannot explicitly deny access, only grant permissions.

3.3.2. Assigning Rights

Rights detail the specific operations a user can perform on directory data. You can allow or deny all rights, or you can assign one or more of the following rights:

Right	Description
Read	Indicates whether users can read directory data. This permission applies only to the search operation.
Write	Indicates whether users can modify an entry by adding, modifying, or deleting <i>attributes</i> . This permission applies to the modify and modrdn operations.
Add	Indicates whether users can create an <i>entry</i> . This permission applies only to the add operation.
Delete	Indicates whether users can delete an <i>entry</i> . This permission applies only to the delete operation.
Search	Indicates whether users can search for the directory data. Users must have Search and Read rights in order to view the data returned as part of a search result. This permission applies only to the search operation.
Compare	Indicates whether the users can compare data they supply with data stored in the directory. With compare rights, the directory returns a success or failure message in response to an inquiry, but the user cannot see the value of the entry or attribute. This permission applies only to the compare

Right	Description
	operation.
Selfwrite	Indicates whether users can add or delete their own DN from a group. This right is used only for group management.
Proxy	Indicates whether the specified DN can access the target with the rights of another entry.
All	Indicates that the specified DN has all rights (read, write, search, delete, compare, and selfwrite) to the targeted entry, <i>excluding</i> proxy rights.

Table 6.2. User Rights

Rights are granted independently of one another. This means, for example, that a user who is granted add rights can create an entry but cannot delete it if delete rights have not been specifically granted. Therefore, when planning the access control policy for your directory, you must ensure that you grant rights in a way that makes sense for users. For example, it does not usually make sense to grant write permission without granting read and search permissions.



NOTE

The proxy mechanism is very powerful and must be used sparingly. Proxy rights are granted within the scope of the ACL, and there is no way to restrict who an entry that has the proxy right can impersonate; that is, when you grant a user proxy rights, that user has the ability to proxy for any user under the target; there is no way to restrict the proxy rights to only certain users. For example, if an entity has proxy rights to the `dc=example,dc=com` tree, that entity can do anything. Make sure you set the proxy ACL at the lowest possible level of the DIT; see [Section 9.11, “Proxied Authorization ACL Example”](#).

3.3.3. Rights Required for LDAP Operations

This section describes the rights you need to grant to users depending on the type of LDAP operation you want to authorize them to perform.

- Adding an entry:
 - Grant add permission on the entry being added.

- Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- Deleting an entry:
 - Grant delete permission on the entry to be deleted.
 - Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- Modifying an attribute in an entry:
 - Grant write permission on the attribute type.
 - Grant write permission on the value of each attribute type. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- Modifying the RDN of an entry:
 - Grant write permission on the entry.
 - Grant write permission on the attribute type used in the new RDN.
 - Grant write permission on the attribute type used in the old RDN, if you want to grant the right to delete the old RDN.
 - Grant write permission on the value of attribute type used in the new RDN. This right is granted by default but could be restricted using the `targattrfilters` keyword.
- Comparing the value of an attribute:
 - Grant compare permission on the attribute type.
- Searching for entries:
 - Grant search permission on each attribute type used in the search filter.
 - Grant read permission on attribute types used in the entry.

The permissions granted on individual attributes or entries can affect a broad range of actions; for example, there are several different permissions users must have to search the directory like the following `ldapsearch` operation:

```
ldapsearch -h host -s base -b "uid=bkolics,dc=example,dc=com" objectclass=*
mail
```

The following ACI is used to determine whether user `bkolics` can be granted access:

```
aci: (targetattr = "mail")(version 3.0; acl "self access to
mail"; allow (read, search) userdn = "ldap:///self";)
```

The search result list is empty because this ACI does not grant access to the *objectclass* attribute. If you want the search operation described above to be successful, modify the ACI to allow read and search access for the *mail* and *objectclass* attributes.

```
aci: (targetattr = "mail || objectclass")(version 3.0; acl "self
  access to mail"; allow (read, search) userdn = "ldap:///self";)
```

3.3.4. Permissions Syntax

In an ACI statement, the syntax for permissions is `allow|deny (rights)`. *rights* is a list of 1 to 8 comma-separated keywords enclosed within parentheses. Valid keywords are `read`, `write`, `add`, `delete`, `search`, `compare`, `selfwrite`, `proxy`, or `all`.

In the following example, `read`, `search`, and `compare` access is allowed, provided the `bind` rule is evaluated to be true:

```
aci: (target="ldap:///dc=example,dc=com") (version 3.0;acl "example";
  allow (read, search, compare) bind_rule;)
```

3.3.5. Access Control and the `modrdn` Operation

To explicitly deny `modrdn` rights using ACIs, target the relevant entries but omit the `targetattr` keyword. For example, to prevent the `cn=helpDeskGroup,ou=groups,o=example.com` group from renaming any entries in the set specified by the pattern `cn=*,ou=people,o=example.com`, add the following ACI:

```
aci: (target="ldap:///cn=*,ou=people,o=example.com")
  (version 3.0; acl "Deny modrdn rights to the helpDeskGroup";
  deny(write)
  groupdn="ldap:///cn=helpDeskGroup,ou=groups,o=example.com";)
```

4. Bind Rules

Depending on the ACIs defined for the directory, for certain operations, you need to *bind* to the directory. *Binding* means logging in or authenticating yourself to the directory by providing credentials (a bind DN and password for SASL or a client certificate for SSL). The credentials provided in the bind operation and the circumstances of the bind determine whether access to the directory is allowed or denied.

Every permission set in an ACI has a corresponding bind rule that details the required credentials and bind parameters.

Bind rules can be simple, such as stating that the person accessing the directory must belong to a specific group. Bind rules can also be more complex, such as requiring that a person must belong to a specific group, must log in from a machine with a specific IP address, and is restricted to access between 8 a.m. and 5 p.m.

Bind rules define who can access the directory, when, and from where by defining any of the following:

- Users, groups, and roles that are granted access.
- Locations from which an entity must bind.
- Times or days on which binding must occur.
- Types of authentication that must be in use during binding.

Additionally, bind rules can be complex constructions that combine these criteria by using Boolean operators. See [Section 4.10, “Using Boolean Bind Rules”](#) for more information.

4.1. Bind Rule Syntax

Whether access is allowed or denied depends on whether an ACI's bind rule is evaluated to be true. Bind rules use one of the two following patterns:

```
keyword = "expression"; or keyword != "expression";
```

Equal (=) indicates that *keyword* and *expression* must match in order for the bind rule to be true, and not equal (!=) indicates that *keyword* and *expression* must not match in order for the bind rule to be true.



NOTE

The `timeofday` keyword also supports the inequality expressions (<, <=, >, >=). This is the only keyword that supports these expressions.

The quotation marks (") around *expression* and the delimiting semicolon (;) are required. The expressions you can use depend on the associated *keyword*.

[Table 6.3, “LDIF Bind Rule Keywords”](#) lists each keyword and the associated expressions and indicates whether wildcard characters are allowed in the expression.

Keyword	Valid Expressions	Wildcard Allowed
userdn	ldap:///distinguished_name ldap:///all ldap:///anyone ldap:///self ldap:///parent ldap:///suffix??scope?(filter)	Yes, in DN only

Keyword	Valid Expressions	Wildcard Allowed
groupdn	ldap:///DN DN	No
roledn	ldap:///DN DN	No
userattr	attribute#bindType orattribute#value	No
ip	IP_address	Yes
dns	DNS_host_name	Yes
dayofweek	sun mon tue wed thu fri sat	No
timeofday	0 - 2359	No
authmethod	none simple ssl sasl sasl_mechanism	No

Table 6.3. LDIF Bind Rule Keywords

4.2. Defining User Access - userdn Keyword

User access is defined using the `userdn` keyword. The `userdn` keyword requires one or more valid distinguished names in the following format:

```
userdn = "ldap:///dn [| ldap:///dn]...[| ldap:///dn]"
```

dn can be a DN or one of the expressions `anyone`, `all`, `self`, or `parent`:

```
userdn = "ldap:///anyone" Defines anonymous access
userdn = "ldap:///all" Defines general access
userdn =ldap:///self" Defines self access
userdn =ldap:///parent" Defines access for the parent entry
```

The `userdn` keyword can also be expressed as an LDAP filter:

```
ldap:///suffix??scope?(filter)
```



NOTE

If a DN contains a comma, the comma must be preceded by a backslash (\) escape character.

4.2.1. Anonymous Access (anyone Keyword)

Granting anonymous access to the directory means that anyone can access it without providing a bind DN or password and regardless of the circumstances of the bind. You can limit anonymous access to specific types of access (for example, read or search access) or to specific subtrees or individual entries within the directory.

From the Directory Server Console, you define anonymous access through the **Access Control Editor**. See [Section 5, "Creating ACLs from the Console"](#).

4.2.2. General Access (all Keyword)

You can use bind rules to indicate that a permission applies to anyone who has successfully bound to the directory; that is, all authenticated users. This allows general access while preventing anonymous access.

From the Directory Server Console, you define general access on the **Access Control Editor**. For more information, see [Section 5, "Creating ACLs from the Console"](#).

4.2.3. Self Access (self Keyword)

Specifies that users are granted or denied access to their own entries. In this case, access is granted or denied if the bind DN matches the DN of the targeted entry.

From the Directory Server Console, you set up self access on the **Access Control Editor**. For more information, see [Section 5, "Creating ACLs from the Console"](#).

4.2.4. Parent Access (parent Keyword)

Specifies that users are granted or denied access to the entry only if their bind DN is the parent of the targeted entry.

You cannot set up parent access control using the Directory Server Console.

4.2.5. LDAP URLs

You can dynamically target users in ACLs using a URL with an LDAP filter:

```
userdn = "ldap:///suffix??scope?(filter)"
```

For example, all users in the accounting and engineering branches of the `example.com` tree would be granted or denied access to the targeted resource dynamically based on the following URL:

```
userdn = "ldap:///dc=example,dc=com??sub?(|(ou=engineering)(ou=accounting))"
```



NOTE

Do not specify a hostname or port number within the LDAP URL. LDAP URLs always apply to the local server.

For more information about LDAP URLs, see [Appendix C, LDAP URLs](#).

4.2.6. Wildcards

You can also specify a set of users by using the wildcard character (*). For example, specifying a user DN of `uid=u*,dc=example,dc=com` indicates that only users with a bind DN beginning with the letter `u` are allowed or denied access based on the permissions you set.

From the Directory Server Console, you set user access from the **Access Control Editor**. For more information, see [Section 5, “Creating ACLs from the Console”](#).

4.2.7. Examples

Example	Description
Userdn = "ldap:///uid=*,dc=example,dc=com"; keyword containing an LDAP URL	The bind rule is evaluated to be true if the user binds to the directory using any distinguished name of the specified pattern. For example, both of the following bind DN's would be evaluated to be true: uid=ssarette,dc=example,dc=com uid=tjaz,ou=Accounting,dc=example,dc=com This bind DN would be evaluated to be false: cn=Babs Jensen,dc=example,dc=com
Userdn="ldap:///uid=bj,dc=example,dc=com ldap:///uid=kc,dc=example,dc=com"; keyword containing logical OR of LDAP URLs	The bind rule is evaluated to be true if the client binds as either of the two supplied distinguished names.
Userdn != ldap:///uid=*,ou=Accounting,dc=example,dc=com; keyword excluding a specific LDAP URL	The bind rule is evaluated to be true if the client is not binding as a UID-based distinguished name in the accounting subtree. This bind rule only makes sense if the targeted entry is not under the accounting branch of the directory tree.

Example	Description
<p><code>userdn = "ldap:///self";</code></p> <p>keyword containing self keyword</p>	<p>The bind rule is evaluated to be true if the user is accessing the entry represented by the DN with which the user bound to the directory. That is, if the user has bound as <code>uid=ssarette,dc=example,dc=com</code> and the user is attempting an operation on the <code>uid=ssarette,dc=example,dc=com</code> entry, then the bind rule is true.</p> <p>If you want to grant all users in the <code>example.com</code> tree write access to their <code>userPassword</code> attribute, you would create the following ACI on the <code>dc=example,dc=com</code> node.</p> <p><code>aci: (targetattr = "userPassword") (version 3.0; acl "write-self"; allow (write) userdn = "ldap:///self");</code></p>
<p><code>userdn = "ldap:///all";</code></p> <p>keyword containing the all keyword</p>	<p>The bind rule is evaluated to be true for any valid bind DN. To be true, a valid distinguished name must be presented by the user for a successful bind operation.</p> <p>For example, if you want to grant read access to the entire tree to all authenticated users, you would create the following ACI on the <code>dc=example,dc=com</code> node:</p> <p><code>aci:(version 3.0; acl "all-read"; allow (read) userdn="ldap:///all");</code></p>
<p><code>userdn = "ldap:///anyone";</code></p> <p>keyword containing the anyone keyword</p>	<p>The bind rule is evaluated to be true for anyone; use this keyword to provide anonymous access to your directory.</p> <p>For example, if you want to allow anonymous read and search access to the entire <code>example.com</code> tree, you would create the following ACI on the <code>dc=example,dc=com</code> node:</p> <p><code>aci: (version 3.0; acl "anonymous-read-search"; allow (read,search) userdn = "ldap:///anyone");</code></p>
<p><code>userdn = "ldap:///parent";</code></p> <p>keyword containing the parent keyword</p>	<p>The bind rule is evaluated to be true if the bind DN is the parent of the targeted entry.</p> <p>For example, if you want to grant write access to every user's child entries, you would create the following ACI on the <code>dc=example,dc=com</code> node:</p> <p><code>aci:(version 3.0; acl "parent access"; allow (write) userdn="ldap:///parent");</code></p>

Table 6.4. userdn Keyword Examples


4.3. Defining Group Access - groupdn Keyword

Members of a specific group can access a targeted resource. This is known as *group access*. Group access is defined using the `groupdn` keyword to specify that access to a targeted entry is granted or denied if the user binds using a DN that belongs to a specific group.

The `groupdn` keyword requires one or more valid distinguished names in the following format:

```
groupdn="ldap:///dn [| ldap:///dn]... [| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the named group.



NOTE

If a DN contains a comma, the comma must be escaped by a backslash (\).

From the Directory Server Console, you can define specific groups using the **Access Control Editor**. For more information, see [Section 5, “Creating ACIs from the Console”](#).

ScExample	Description
<code>groupdn = key ldap:///cn=Administrators,dc=example,dc=com containing an LDAP URL</code>	The bind rule is evaluated to be true if the bind DN belongs to the Administrators group. If you wanted to grant the Administrators group permission to write to the entire directory tree, you would create the following ACI on the <code>dc=example,dc=com</code> node: aci: (version 3.0; acl "Administrators-write"; allow (write) groupdn="ldap:///cn=Administrators,dc=example,dc=com");
<code>groupdn = key ldap:///cn=Administrators,dc=example,dc=com or key ldap:///cn=Mail Administrators,dc=example,dc=com"; OR of LDAP URLs</code>	The bind rule is evaluated to be true if the bind DN belongs to either the Administrators or the Mail Administrators group.

Table 6.5. groupdn Examples

4.4. Defining Role Access - roledn Keyword

Members of a specific role can access a targeted resource. This is known as *role access*. Role access is defined using the `roledn` keyword to specify that access to a targeted entry is granted

or denied if the user binds using a DN that belongs to a specific role.

The `roledn` keyword requires one or more valid distinguished names in the following format :

```
roledn = "ldap:///dn [| ldap:///dn]... [| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the specified role.



NOTE

If a DN contains a comma, the comma must be escaped by a backslash (\).

The `roledn` keyword has the same syntax and is used in the same way as the `groupdn` keyword.

4.5. Defining Access Based on Value Matching

You can set bind rules to specify that an attribute value of the entry used to bind to the directory must match an attribute value of the targeted entry.

For example, you can specify that the bind DN must match the DN in the *manager* attribute of a user entry in order for the ACI to apply. In this case, only the user's manager would have access to the entry.

This example is based on DN matching. However, you can match any attribute of the entry used in the bind with the targeted entry. For example, you could create an ACI that allowed any user whose *favoriteDrink* attribute is *beer* to read all the entries of other users that have the same value for *favoriteDrink*.

4.5.1. Using the `userattr` Keyword

The `userattr` keyword can be used to specify which attribute values must match between the entry used to bind and the targeted entry. You can specify any of the following:

- A user DN
- A group DN
- A role DN
- An LDAP filter, in an LDAP URL
- Any attribute type

The LDIF syntax of the `userattr` keyword is as follows:

```
userattr = "attrName#bindType"
```

Using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter has the following format:

```
userattr = "attrName#attrValue"
```

- *attrName* is the name of the attribute used for value matching.
- *bindType* is either `USERDN`, `GROUPDN`, or `LDAPURL`.
- *attrValue* is any string representing an attribute value.

4.5.1.1. Example with USERDN Bind Type

The following associates the `userattr` keyword with a bind based on the user DN:

```
userattr = "manager#USERDN"
```

The bind rule is evaluated to be true if the bind DN matches the value of the *manager* attribute in the targeted entry. You can use this to allow a user's manager to modify employees' attributes. This mechanism only works if the *manager* attribute in the targeted entry is expressed as a full DN.

The following example grants a manager full access to his or her employees' entries:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)
      (version 3.0; acl "manager-write"; allow (all) userattr =
      "manager#USERDN";)
```

4.5.1.2. Example with GROUPDN Bind Type

The following associates the `userattr` keyword with a bind based on a group DN:

```
userattr = "owner#GROUPDN"
```

The bind rule is evaluated to be true if the bind DN is a member of the group specified in the *owner* attribute of the targeted entry. For example, you can use this mechanism to allow a group to manage employees' status information. You can use an attribute other than *owner* as long as the attribute you use contains the DN of a group entry.

The group you point to can be a dynamic group, and the DN of the group can be under any suffix in the database. However, the evaluation of this type of ACI by the server is very resource

intensive.

If you are using static groups that are under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?owner#GROUPDN"
```

In this example, the group entry is under the `dc=example,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

(By default, `owner` is not an allowed entry in a user's entry. You would have to extend your schema to allow this attribute in a `person` object.)

4.5.1.3. Example with ROLEDN Bind Type

The following associates the `userattr` keyword with a bind based on a role DN:

```
userattr = "exampleEmployeeReportsTo#ROLEDN"
```

The bind rule is evaluated to be true if the bind DN belongs to the role specified in the `exampleEmployeeReportsTo` attribute of the targeted entry. For example, if you create a nested role for all managers in your company, you can use this mechanism to grant managers at all levels access to information about employees that are at a lower grade than themselves.



NOTE

This example assumes that you have added the `exampleEmployeeReportsTo` attribute to the schema and that all employee entries contain this attribute. It also assumes that the value of this attribute is the DN of a role entry. For information on adding attributes to the schema, see [Section 2.2, "Creating Attributes"](#).

The DN of the 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, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?employeeReportsTo#ROLEDN"
```

In this example, the role entry is under the `dc=example,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

4.5.1.4. Example with LDAPURL Bind Type

The following associates the `userattr` keyword with a bind based on an LDAP filter:

```
userattr = "myfilter#LDAPURL"
```

The bind rule is evaluated to be true if the bind DN matches the filter specified in the *myfilter* attribute of the targeted entry. The *myfilter* attribute can be replaced by any attribute that contains an LDAP filter.

4.5.1.5. Example with Any Attribute Value

The following associates the `userattr` keyword with a bind based on any attribute value:

```
userattr = "favoriteDrink#Beer"
```

The bind rule is evaluated to be true if the bind DN and the target DN include the *favoriteDrink* attribute with a value of *Beer*.

4.5.1.6. 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 some circumstances, you might want to extend the application of the ACI several levels below the targeted entry. This is possible by using the `parent` keyword and specifying the number of levels below the target that should inherit the ACI.

When you use the `userattr` keyword in association with the `parent` keyword, the syntax is as follows:

```
userattr = "parent[inheritance_level].attrName#bindType"
```

Using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter, the syntax is as follows:

```
userattr = "parent[inheritance_level].attrName#attrValue"
```

- *inheritance_level* is a comma-separated list that indicates how many levels below the target inherits the ACI. You can include five levels (0, 1, 2, 3, 4) below the targeted entry; zero (0) indicates the targeted entry.
- *attribute* is the attribute targeted by the `userattr` or `groupattr` keyword.
- *bindType* can be one of `USERDN`, `GROUPDN`, or `LDAPURL`.

For example:

```
userattr = "parent[0,1].manager#USERDN"
```

This bind rule is evaluated to be true if the 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.

The example in [Figure 6.1, “Using Inheritance With the userattr Keyword”](#) indicates that user `bjensen` is allowed to read and search the `cn=Profiles` entry as well as the first level of child entries which includes `cn=mail` and `cn=news`, thus allowing her to search through her own mail and news IDs.

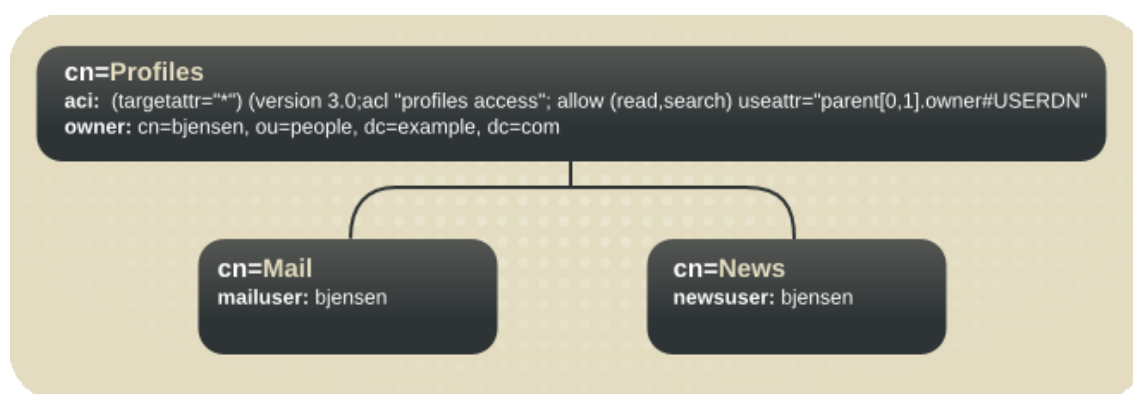


Figure 6.1. Using Inheritance With the userattr Keyword

In this example, if you did not use inheritance, you would have to do one of the following to achieve the same result:

- Explicitly set read and search access for user `bjensen` on the `cn=Profiles`, `cn=mail`, and `cn=news` entries in the directory.
- Add the owner attribute with a value of `bjensen` to the `cn=mail` and `cn=news` entries, and then add the following ACI to the `cn=mail` and `cn=news` entries.

```
aci: (targetattr=*)" (version 3.0; acl "profiles access"; allow
(read,search)
  userattr="owner#USERDN" ; )
```

4.5.1.7. Granting Add Permission Using the userattr Keyword

Using the `userattr` keyword in conjunction with `all` or `add` permissions does not behave as one would typically expect. Typically, when a new entry is created in the directory, Directory Server evaluates access rights on the entry being created and not on the parent entry. However, in the case of ACIs using the `userattr` keyword, this behavior could create a security

hole, and the server's normal behavior is modified to avoid it.

Consider the following example:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*) (version 3.0;  
acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

This ACI grants managers all rights on the entries of employees that report to them. However, because access rights are evaluated on the entry being created, this type of ACI would also allow any employee to create an entry in which the manager attribute is set to their own DN. For example, disgruntled employee Joe (`cn=Joe,ou=eng,dc=example,dc=com`) might want to create an entry in the Human Resources branch of the tree to use (or misuse) the privileges granted to Human Resources employees.

He could do this by creating the following entry:

```
dn: cn= Trojan Horse,ou=Human Resources,dc=example,dc=com  
objectclass: top  
...  
cn: Trojan Horse  
manager: cn=Joe,ou=eng,dc=example,dc=com
```

To avoid this type of security threat, the ACI evaluation process does not grant add permission at level 0, to the entry itself. You can, however, use the `parent` keyword to grant add rights below existing entries. You must specify the number of levels below the parent for add rights. For example, the following ACI allows child entries to be added to any entry in the `dc=example,dc=com` that has a *manager* attribute that matches the bind DN:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)  
(version 3.0; acl "parent-access"; allow (add)  
userattr = "parent[0,1].manager#USERDN";)
```

This ACI ensures that add permission is granted only to users whose bind DN matches the manager attribute of the parent entry.

4.6. Defining Access from a Specific IP Address

Using bind rules, you can indicate that the bind operation must originate from a specific IP address. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on an IP address is as follows:

```
ip = "IP_address" or ip != "IP_address"
```

The IP address must be expressed in dot notation. You can use the wildcard character (*) to include multiple machines. For example, the following string is valid:


```
ip = "12.123.1.*";
```

The `bind` rule is evaluated to be true if the client accessing the directory is located at the named IP address. This can be useful for allowing certain kinds of directory access only from a specific subnet or machine.

For example, use a wildcard IP address such as `12.3.45.*` to specify a specific subnet or `123.45.6.*+255.255.255.115` to specify a subnet mask.

From the Directory Server Console, you can define specific machines to which the ACI applies through the **Access Control Editor**. For more information, see [Section 5, “Creating ACIs from the Console”](#).

4.7. Defining Access from a Specific Domain

A `bind` rule can specify that the bind operation must originate from a particular domain or host machine. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a `bind` rule based on the DNS hostname is as follows:

```
dns = "DNS_Hostname or dns != "DNS_Hostname
```



CAUTION

The `dns` keyword requires that the naming service used on your machine is DNS. If the name service is not DNS, use the `ip` keyword instead.

The `dns` keyword requires a fully qualified DNS domain name. Granting access to a host without specifying the domain creates a potential security threat. For example, the following expression is allowed but not recommended:

```
dns = "legend.eng";
```

Instead, use a fully qualified name:

```
dns = "legend.eng.example.com";
```

The `dns` keyword allows wildcards. For example:

```
dns = "*.example.com";
```

The bind rule is evaluated to be true if the client accessing the directory is located in the named domain. This can be useful for allowing access only from a specific domain. Wildcards will not work if your system uses a naming service other than DNS. In such a case, if you want to restrict access to a particular domain, use the `ip` keyword, as described in [Section 4.6](#), “*Defining Access from a Specific IP Address*”.

4.8. Defining Access at a Specific Time of Day or Day of Week

You can use bind rules to specify that binding can only occur at a certain time of day or on a certain day of the week. For example, you can set a rule that allows access only if it is between the hours of 8 a.m. and 5 p.m. Monday through Friday. The time used to evaluate access rights is the time on the Directory Server, not the time on the client.

The LDIF syntax for setting a bind rule based on the time of day is as follows:

```
timeofday operator time
```

operator can be one of the following symbols:

equal to (=)

not equal to (!=)

greater than (>)

greater than or equal to (>=)

less than (<)

less than or equal to (<=)

The `timeofday` keyword requires a time of day expressed in hours and minutes in the 24 hour clock (0 to 2359).



NOTE

The time on the Directory Server is used for the evaluation, not the time on the client.

The LDIF syntax for setting a bind rule based on the day in the week is as follows:

```
dayofweek = "day1, day2 ..."
```

The possible values for the `dayofweek` keyword are the English three-letter abbreviations for the days of the week: `sun`, `mon`, `tue`, `wed`, `thu`, `fri`, `sat`.

4.8.1. Examples

The following are examples of the `timeofday` and `dayofweek` syntax:

- The bind rule is evaluated to be true if the client is accessing the directory at noon.

```
timeofday = "1200";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time other than 1 a.m.

```
timeofday != "0100";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time after 8 a.m.

```
timeofday > "0800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time before 6 p.m.

```
timeofday < "1800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at 8 a.m. or later.

```
timeofday >= "0800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at 6 p.m. or earlier.

```
timeofday <= "1800";
```

- The bind rule is evaluated to be true if the client is accessing the directory on Sunday, Monday, or Tuesday.

```
dayofweek = "Sun, Mon, Tue";
```

4.9. Defining Access Based on Authentication Method

You can set bind rules that state that a client must bind to the directory using a specific authentication method. There are four available authentication methods:

- *None*. Authentication is not required. This is the default. It represents anonymous access.
-

Simple. The client must provide a user name and password to bind to the directory.

- *SSL.* The client must bind to the directory over a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection, using a client certificate for authentication.

In the case of SSL, the connection is established to the LDAPS second port; in the case of TLS, the connection is established through a Start TLS operation. In both cases, a certificate must be provided. For information on setting up SSL, see [Chapter 11, Managing SSL](#).

- *SASL.* The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. Directory Server supports three SASL mechanisms: `EXTERNAL`, `CRAM-MD5`, `DIGEST-MD5`, and `GSS-API` (for Kerberos systems). For information on setting up SASL, see [Chapter 12, Managing SASL](#).



NOTE

You cannot set up authentication-based bind rules through the **Access Control Editor**.

The LDIF syntax for setting a bind rule based on an authentication method is as follows:

```
authmethod = "sasl_mechanism"
```

sasl_mechanism can be `none`, `simple`, `ssl`, or `"sasl sasl_mechanism"`.

4.9.1. Examples

The following are examples of the `authmethod` keyword:

- Authentication is not checked during bind rule evaluation.

```
authmethod = "none";
```

- The bind rule is evaluated to be true if the client is accessing the directory using a username and password.

```
authmethod = "simple";
```

- The bind rule is evaluated to be true if the client authenticates to the directory using a certificate over LDAPS. This is not evaluated to be true if the client authenticates using simple

authentication (bind DN and password) over LDAPS.

```
authmethod = "ssl";
```

- The bind rule is evaluated to be true if the client is accessing the directory using the SASL DIGEST-MD5 mechanism.

```
authmethod = "sasl DIGEST-MD5";
```

4.10. Using Boolean Bind Rules

Bind rules can be complex expressions that use the Boolean expressions `AND`, `OR`, and `NOT` to set very precise access rules. You cannot use the Directory Server Console to create Boolean bind rules. You must create an LDIF statement.

The LDIF syntax for a Boolean bind rule is as follows:

```
bind_rule [boolean][bind_rule][boolean][bind_rule]...;
```

For example, this bind rule is evaluated to be true if the bind DN is a member of either the administrator's group or the Mail Administrator's group and if the client is running from within the `example.com` domain:

```
(groupdn = "ldap:///cn=administrators,dc=example,dc=com" or
 groupdn = "ldap:///cn=mail administrators,dc=example,dc=com" and
 dns = "/*.example.com");
```

The trailing semicolon (;) is a required delimiter that must appear after the final bind rule.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.
- `NOT` before `AND` or `OR` operators.
- `OR` and `AND` operators have no order of precedence.

Consider the following Boolean bind rules:

```
(bind_rule_A) OR (bind_rule_B)
(bind_rule_B) OR (bind_rule_A)
```

Because Boolean expressions are evaluated from left to right, in the first case, bind rule A is evaluated before bind rule B, and, in the second case, bind rule B is evaluated before bind rule A.

However, the Boolean `NOT` is evaluated *before* the Boolean `OR` and Boolean `AND`. Thus, in the following example, bind rule B is evaluated before bind rule A despite the left-to-right rule.

```
(bind_rule_A) AND NOT (bind_rule_B)
```

5. Creating ACIs from the Console

You can use the Directory Server Console to view, create, edit, and delete access control instructions for your directory:

- [Section 5.1, “Displaying the Access Control Editor”](#)
- [Section 5.2, “Creating a New ACI”](#)
- [Section 5.3, “Editing an ACI”](#)
- [Section 5.4, “Deleting an ACI”](#)

See [Section 9, “Access Control Usage Examples”](#) for a collection of access control rules commonly used in Directory Server security policies, along with step-by-step instructions for using the Directory Server Console to create them.

The **Access Control Editor** prevents creating more complex ACIs in visual editing mode, especially ACIs with any of these characteristics:

- Deny access ([Section 3.3.4, “Permissions Syntax”](#)).
- Create value-based ACIs ([Section 3.2.2, “Targeting Attributes”](#)).
- Define parent access ([Section 4.2.4, “Parent Access \(parent Keyword\)”](#)).
- Create ACIs that contain Boolean bind rules ([Section 4.10, “Using Boolean Bind Rules”](#)).
- Create ACIs that use the `roledn`, `userattr`, `authmethod` keywords.



TIP

In the **Access Control Editor**, click the **Edit Manually** button at any time to check the LDIF representation of the ACI changes made through the graphical interface.

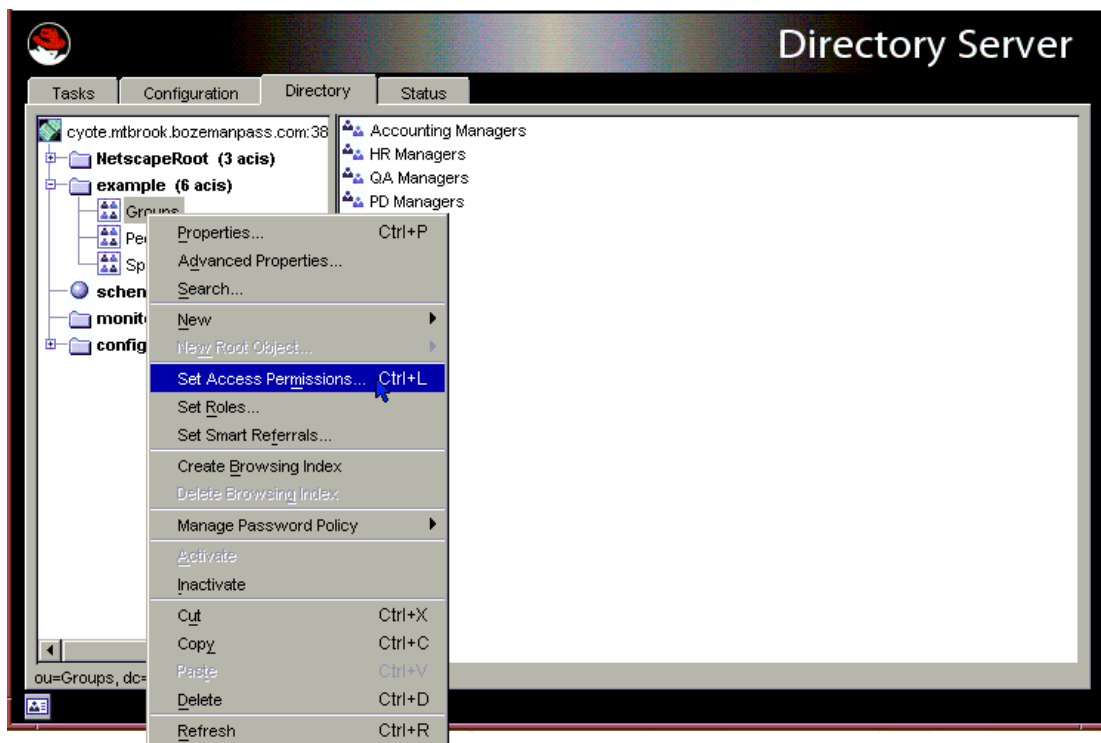
5.1. Displaying the Access Control Editor

1. Start the Directory Server Console. Log in using the bind DN and password of a privileged user, such as the Directory Manager, who has write access to the ACIs configured for the directory.

```
/usr/bin/redhat-idm-console
```

2. Select the **Directory** tab.
3. Right-click the entry in the navigation tree for which to set access control, and select **Set Access Permissions** from the pop-up menu.

Alternatively, highlight the entry, and select **Set Access Permissions** from the **Object** menu.



4. Click **New** to open the **Access Control Editor**.

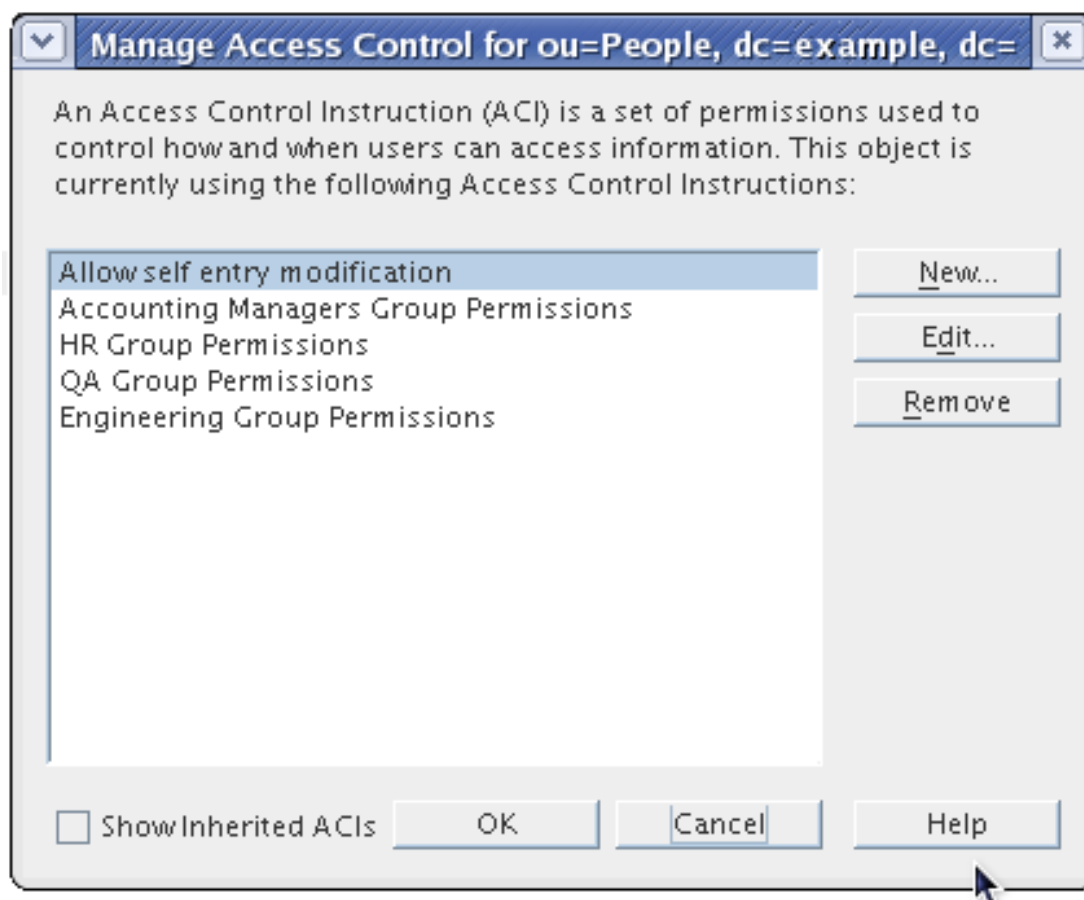


Figure 6.2. Access Control Editor Window

5.2. Creating a New ACI

To create a new ACI in the Directory Server Console, do the following:

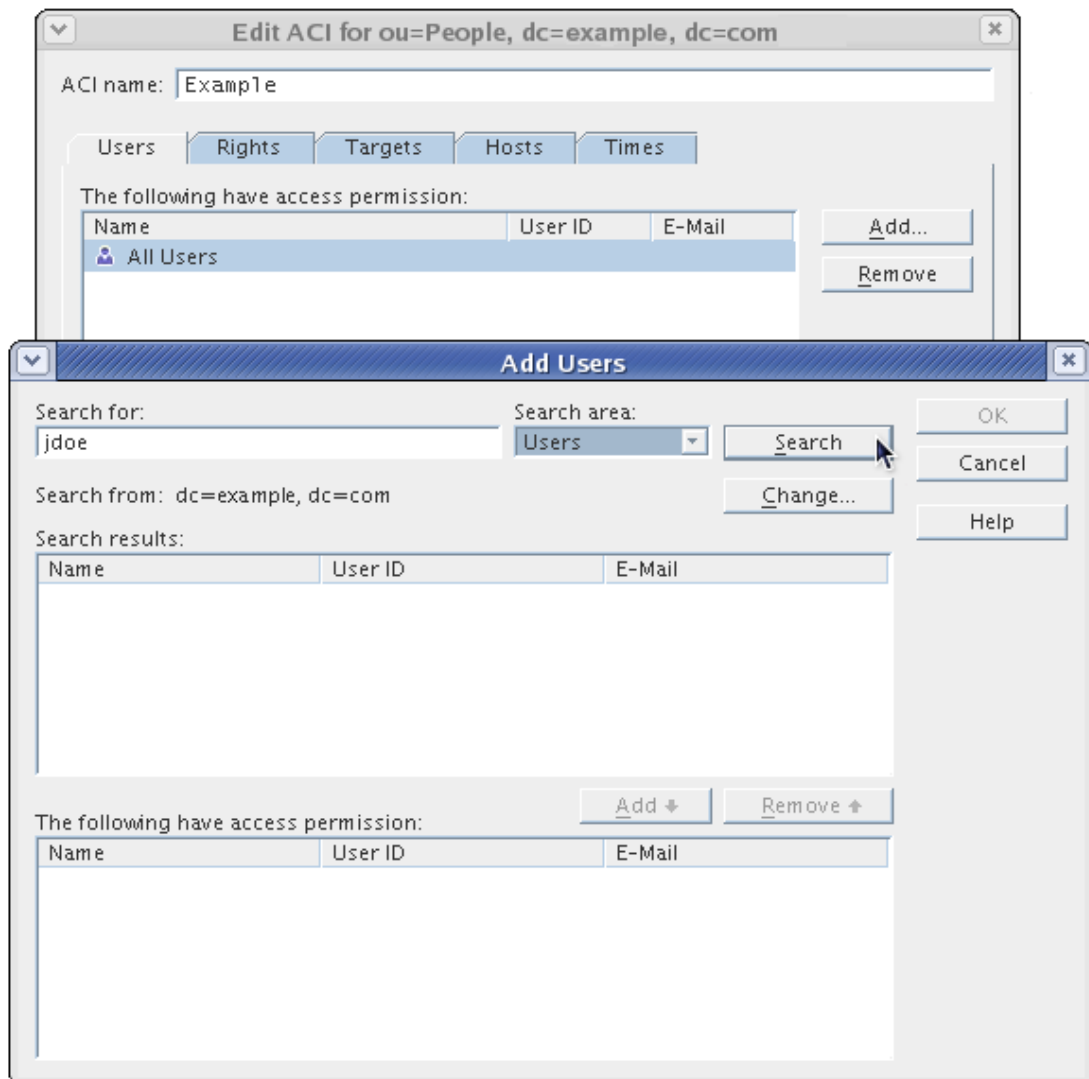
1. Open the **Access Control Editor**, as described in [Section 5.1, “Displaying the Access Control Editor”](#).

If the view displayed is different from [Figure 6.2, “Access Control Editor Window”](#), click the **Edit Visually** button.

2. Type the ACI name in the **ACI Name** field.

The name can be any unique string to identify the ACI. If you do not enter a name, the server uses `unnamed ACI`.

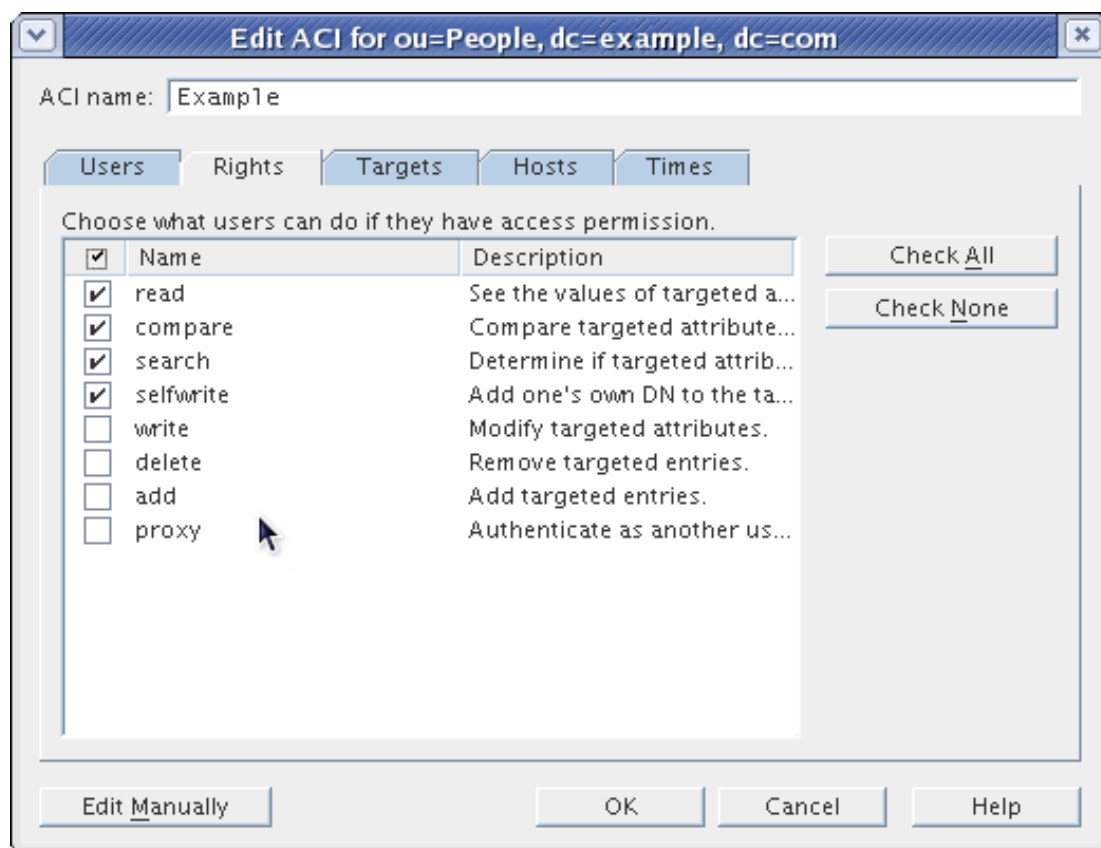
3. In the **Users/Groups** tab, select the users to whom you are granting access by highlighting **All Users** or clicking the **Add** button to search the directory for the users to add.



- a. Select a search area from the drop-down list, enter a search string in the **Search** field, and click the **Search** button. You can use wildcards (an asterisk, *) to search for partial usernames. The search results are displayed in the list below.
- b. Highlight the entries you want in the search result list, and click the **Add** button to add them to the list of entries which have access permission.
- c. Click **OK** to dismiss the **Add Users and Groups** window.

The selected entries are now listed on the **Users/Groups** tab in the ACI editor.

4. In the **Access Control Editor**, click the **Rights** tab, and use the checkboxes to select the rights to grant.



5. Click the **Targets** tab. Click **This Entry** to display the current node as the target for the ACI or click **Browse** to select a different suffix.

ACI name:

Users Rights **Targets** Hosts Times

Target directory entry:

Filter for sub-entries:

These attributes are affected for all entries:

<input checked="" type="checkbox"/>	Name	OID
<input checked="" type="checkbox"/>	nsAdminCgiWaitPid	nsAdminCgiWaitPid-oid
<input checked="" type="checkbox"/>	altServer	1.3.6.1.4.1.1466.101.120.6
<input checked="" type="checkbox"/>	pamExcludeSuffix	2.16.840.1.113730.3.1.2068
<input checked="" type="checkbox"/>	nsDS5ReplicaRoot	2.16.840.1.113730.3.1.584
<input checked="" type="checkbox"/>	nsSNMPEnabled	2.16.840.1.113730.3.1.232
<input checked="" type="checkbox"/>	ntUserCreateNewAccou...	2.16.840.1.113730.3.1.42
<input checked="" type="checkbox"/>	javaCodebase	1.3.6.1.4.1.42.2.27.4.1.7
<input checked="" type="checkbox"/>	nscpEntryDN	2.16.840.1.113730.3.1.545
<input checked="" type="checkbox"/>	nsSSLPersonalitySSL	nsSSLPersonalitySSL-oid



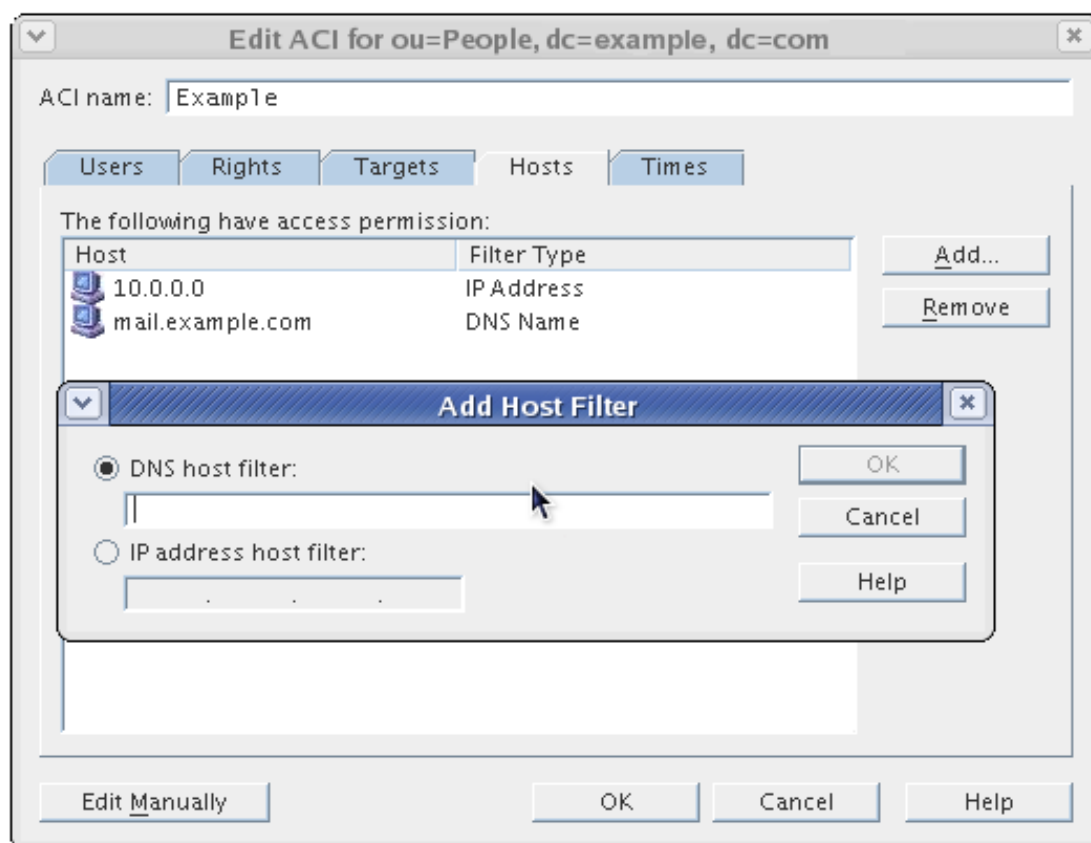
NOTE

You can change the value of the target DN, but the new DN must be a direct or indirect child of the selected entry.

If you do not want every entry in the subtree under this node to be targeted by the ACI, enter a filter in the **Filter for Sub-entries** field. The filter applies to every entry below the target entry; for example, setting a filter of `ou=Sales` means that only entries with `ou=Sales` in their DN are returned.

Additionally, you can restrict the scope of the ACI to only certain attributes by selecting the attributes to target in the attribute list.

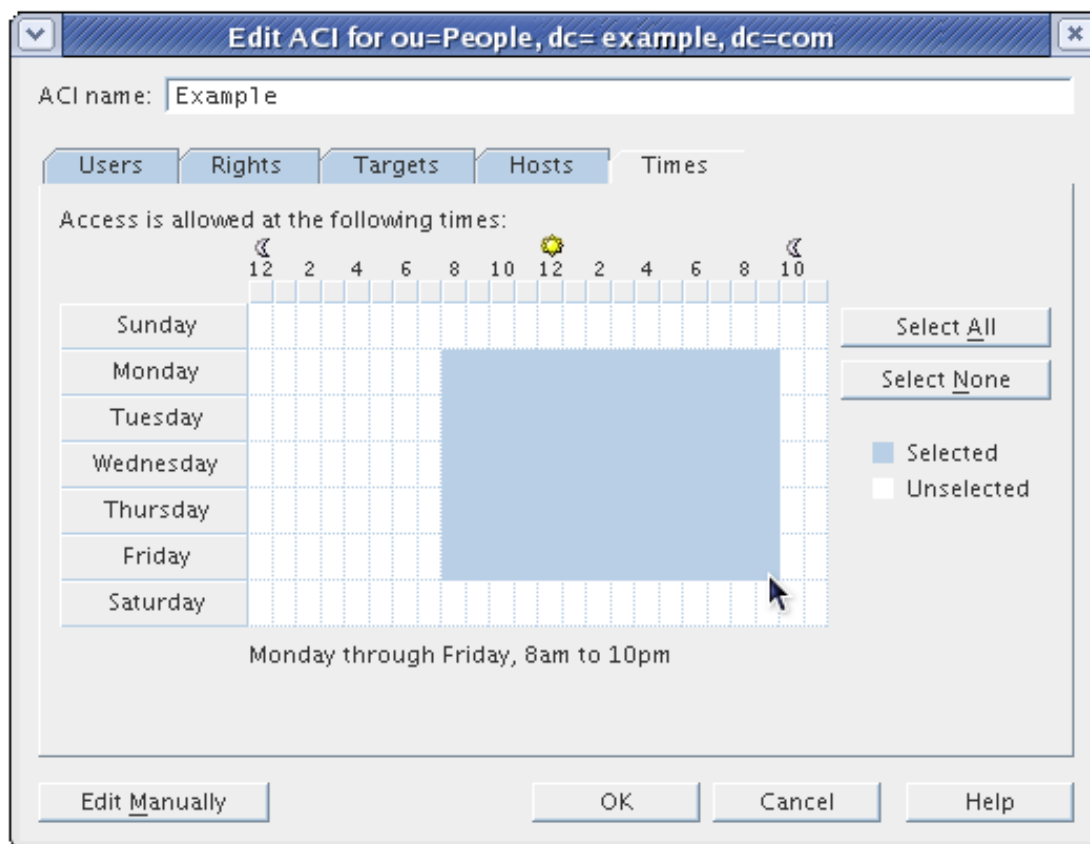
- Click the **Hosts** tab, then the **Add** button to open the **Add Host Filter** dialog box.



You can specify a hostname or an IP address. With an IP address, you can use an asterisk (*) as a wildcard.

7. Click the **Times** tab to display the table showing at what times access is allowed.

By default, access is allowed at all times. You can change the access times by clicking and dragging the cursor over the table. You cannot select discrete blocks of time, only continuous time ranges.



8. When you have finished editing the ACI, click **OK**.

The **Access Control Editor** closes, and the new ACI is listed in the **Access Control Manager** window.



NOTE

For any point of creating the ACI, you can click the **Edit Manually** button to display the LDIF statement corresponding to the wizard input. You can modify this statement, but your changes may not be visible in the graphical interface.

5.3. Editing an ACI

To edit an ACI, do the following:

1. In the **Directory** tab, right-click the top entry in the subtree, and choose **Set Access Permissions** from the pop-up menu.

The **Access Control Manager** window opens, listing the ACIs belonging to the entry.

2. In the **Access Control Manager** window, highlight the ACI to edit, and click **Edit**.
3. Make the edits to the ACI in the **Access Control Editor**; the different screens are described more in [Section 5.2, “Creating a New ACI”](#) and in the online help.
4. When you have finished editing the ACI, click **OK**.

The **Access Control Editor** window closes, and the modified ACI is listed in the **Access Control Manager**.

5.4. Deleting an ACI

To delete an ACI, do the following:

1. In the **Directory** tab, right-click the top entry in the subtree, and choose **Set Access Permissions** from the pop-up menu.

The **Access Control Manager** window opens with a list of ACIs belonging to the entry.

2. In the **Access Control Manager** window, select the ACI to delete.
3. Click **Remove**.

The ACI is no longer listed in the **Access Control Manager** window.

6. Viewing ACIs

All the ACIs under a single suffix in the directory can be viewed from the command line by using the following `ldapsearch` command:¹

```
ldapsearch -h host -p port -b baseDN -D rootDN -w rootPassword (aci=*) aci
```

See the *Directory Server Configuration, Command, and File Reference* for information on using the `ldapsearch` utility.

From the Directory Server Console, all of the ACIs that apply to a particular entry can be viewed through the **Access Control Manager**.

1. Start the Directory Server Console. See [Section 4, “Starting the Directory Server Console”](#).
2. In the **Directory** tab, right-click the entry in the navigation tree, and select **Set Access**

¹ The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

Permissions.

The **Access Control Manager** opens with a list of the ACIs belonging to the selected entry.

3. Check the **Show Inherited ACIs** checkbox to display all ACIs created on entries above the selected entry that also apply.

7. Get Effective Rights Control

Finding the rights on existing attributes within a specific entry offers a convenient way for administrators to find and control the access rights.

Get effective rights is an extended `ldapsearch` which returns the access control permissions set on each attribute within an entry. The effective rights can be retrieved by sending an LDAP control along with a search operation. The results show the effective rights on each returned entry and each attribute of each returned entry.

The access control information is divided into two groups of access: rights for an entry and rights for an attribute. *Rights for an entry* means the rights, such as modify or delete, that are limited to that specific entry. *Rights for an attribute* means the access right to every instance of that attribute throughout the directory.

Some of the situations when this kind of detailed access control may be necessary include the following:

- An administrator can use the get effective rights command for minute access control, such as allowing certain groups or users access to entries and restricting others. For instance, members of the `QA Managers` group may have the right to search and read attributes like `manager` and `salary` but only `HR Group` members have the rights to modify or delete them.
- A user can run the get effective rights command to see what attributes he can view or modify on his personal entry. For instance, a user should have access to attributes such as `homePostalAddress` and `cn` but may only have read access to `manager` and `salary`.

An `ldapsearch` run with the `-J` option (which sets the get effective rights control) returns the access controls placed on a particular entry. The `entryLevelRights` and `attributeLevelRights` returns are added as attributes to the bottom of the query results. If `ldapsearch` is run without `-J`, then the entry information is returned as normal, without the `entryLevelRights` or `attributeLevelRights` information.

A get effective rights result looks like the following:

```
dn: uid=tmorris, ou=People, dc=example,dc=com
l: Santa Clara
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: vadm
attributeLevelRights: l:rscwo, userPassword:wo
```

In this example, Ted Morris has the right to add, view, delete, or rename the DN on his own entry, as shown by the return values in `entryLevelRights`. For attributes, he has the right to read, search, compare, self-modify, or self-delete the location (1) attribute but only self-write and self-delete rights to his password, as shown in the `attributeLevelRights` return value.

Information is not given for attributes in an entry that do not have a value; for example, if the `userPassword` value is removed, then a future effective rights search on the entry above would not return any effective rights for `userPassword`, even though self-write and self-delete rights could be allowed. Likewise, if the `street` attribute were added with read, compare, and search rights, then `street: rsc` would appear in the `attributeLevelRights` results.

[Table 6.6, “Permissions That Can Be Set on Entries”](#) and [Table 6.7, “Permissions That Can Be Set on Attributes”](#) summarize the permissions that can be set on entries and on attributes that are retrieved by the get effective rights operation.

Permission	Description
a	Add.
d	Delete.
n	Rename the DN.
v	View the entry.

Table 6.6. Permissions That Can Be Set on Entries

Permission	Description
r	Read.
s	Search.
w	Write (mod-add).
o	Obliterate(mod-del). Analogous to delete.
c	Compare.
W	Self-write.
O	Self-delete.

Table 6.7. Permissions That Can Be Set on Attributes

7.1. Using Get Effective Rights from the Command-Line

To retrieve the effective rights with `ldapsearch`, you must pass the control information with the `ldapsearch` utility's `-J` option, as follows:

```
ldapsearch -p port -h host -D bindDN -w bindPassword -b search_base
-J control OID:boolean criticality:dn:AuthId
```


- *search_base* specifies the entry or entries being checked, while *AuthId* checks the rights of the *AuthId* entry over the *search_base* entry.
- *control OID* is the OID for the get effective rights control, 1.3.6.1.4.1.42.2.27.9.5.2.
- *boolean criticality* specifies whether the search operation should return an error if the server does not support this control (*true*) or if it should be ignored and let the search return as normal (*false*).
- *AuthId* is the DN of the entry whose rights over the *user* account are being checked. If the *AuthId* is left blank (*dn:*), then the rights of an anonymous user are returned.

A user, such as Ted Morris, can use this `ldapsearch` option to retrieve the rights he has to his personal entry, as shown below. Along with returning the effective rights information, the `ldapsearch` returns the regular entry information:

```
ldapsearch -p 389 -h localhost -D "uid=tmorris,ou=people,dc=example,dc=com"
-w password
  -b "uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:
  dn:uid=tmorris,ou=people,dc=example,dc=com" "(objectClass=*)"

version: 1
dn: uid=tmorris, ou=People, dc=example,dc=com
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller, ou=People, dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: v
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc,
manager:rsc, roomNumber:rscwo, mail:rscwo,
facsimileTelephoneNumber:rscwo, objectClass:rsc, uid:rsc,
cn:rsc, userPassword:wo
```

An administrative user, such as Directory Manager, can use the get effective rights operation to determine what rights are granted between users. The following is a sample `ldapsearch` to retrieve effective rights that a manager, Dave Miller (shown in the *dn: user* part of the *-J* value), has over the entry of one of his subordinates, Ted Morris (shown in the *-b* value):

```
ldapsearch -p 389 -h localhost -D "cn=directory manager" -w password
```

```
-b "uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:dn:
uid=dmiller,ou=people,dc=example,dc=com" "(objectClass=*)"

version: 1
dn: uid=tmorris, ou=People, dc=example,dc=com
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller, ou=People, dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: vadm
attributeLevelRights: givenName:rscwo, sn:rscwo, ou:rscwo,
l:rscwo, manager:rscwo, roomNumber:rscwo, mail:rscwo,
facsimileTelephoneNumber:rscwo, objectClass:rscwo, uid:rscwo,
cn:rscwo, userPassword:rscwo
```

For all attributes, Dave Miller has read, search, compare, modify, and delete permissions to Ted Morris's entry. These results are different than the ones returned in checking Ted Morris's access to his own entry, since he personally had only read, search, and compare rights to most of these attributes.

Only an administrator can retrieve effective rights to another user's entry. If Ted Morris tried to determine Dave Miller's rights to Dave Miller's entry, then he would receive the following error:

```
ldapsearch -p 389 -h localhost -D "uid=dmiller,ou=people,dc=example,dc=com"
-w password
-b "uid=tmorris,ou=people,dc=example,dc=com" -J
"1.3.6.1.4.1.42.2.27.9.5.2:true:dn:
uid=tmorris,ou=people,dc=example,dc=com" "(objectClass=*)"

ldap_search: Insufficient access

ldap_search: additional info: get-effective-rights: requestor has no g
permission on the entry
```

However, Ted Morris could run a get effective rights search on his personal entry to determine the rights another user, such as Sam Carter, has to it. Assuming that an `ldapsearch` was run with `-b` set to `uid=tmorris,ou=people,dc=example,dc=com` and the `AuthId` was set to `uid=scarter,ou=people,dc=example,dc=com`, then Ted Morris would retrieve the following

effective rights information:

```
entryLevelRights: v

attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc, manager:rsc,
roomNumber:rsc, mail:rsc,
facsimileTelephoneNumber:rsc, objectClass:rsc, uid:rsc,
cn:rsc, userPassword:none
```

This means that Sam Carter has the right to view the DN of the entry and to read, search; the right to compare the `ou`, `givenName`, `l`, and other attributes; and no rights to the `userPassword` attribute.

7.2. Using Get Effective Rights from the Console

To view effective rights from the Console, do the following:

1. Open the **Directory** tab, and right-click the entry which rights you want to check.
2. Select **Advanced Properties** from the drop-down menu.

The **Property Editor** appears.

3. Check the **Show effective rights** checkbox.

The attribute-level effective rights (`r`, `s`, `c`, `w`, `o`) appear next to the attributes. The entry-level rights (`v`, `a`, `d`, `n`) appear under the full DN for the entry in the lower left-hand corner of the **Property Editor**.

If you check the **Show all allowed attributes** checkbox, then the effective rights for those attributes appear next to the additional attributes, even though they do not have values.

7.3. Get Effective Rights Return Codes

If the criticality is set to `false` for a get effective rights search and an error occurs, the regular entry information is returned, but, in place of rights for `entryLevelRights` and `attributeLevelRights`, an error code is returned. This code can give information on the configuration of the entry that was queried. [Table 6.8, “Returned Result Codes”](#) summarizes the error codes and the potential configuration information they can relay.

Code	Description
0	Successfully completed.
1	Operation error.
12	The critical extension is unavailable. If the criticality expression is set to <code>true</code> and effective rights do not exist on the entry being queried, then this error is returned.

Code	Description
16	No such attribute. If an attribute is specifically queried for access rights but that attribute does not exist in the schema, this error is returned.
17	Undefined attribute type.
21	Invalid attribute syntax.
50	Insufficient rights.
52	Unavailable.
53	Unwilling to perform.
80	Other.

Table 6.8. Returned Result Codes

8. Logging Access Control Information

To obtain information on access control in the error logs, you must set the appropriate log level. To set the error log level from the Console:

1. In the Console, click the **Directory** tab, right-click the config node, and choose **Properties** from the pop-up menu.

This displays the **Property Editor** for the `cn=config` entry.

2. Scroll down the list of attribute value pairs to locate the **nsslapd-errorlog-level** attribute.
3. Add 128 to the value already displayed in the **nsslapd-errorlog-level** value field.

For example, if the value already displayed is 8192 (replication debugging), change the value to 8320. For complete information on error log levels, see the *Directory Server Configuration, Command, and File Reference*.

4. Click **OK** to dismiss the **Property Editor**.

9. Access Control Usage Examples

The examples provided in this section illustrate how an imaginary ISP company, `example.com`, would implement its access control policy. All the examples explain how to perform a given task from the Console and using an LDIF file.

`example.com`'s business is to offer a web hosting service and Internet access. Part of `example.com`'s web hosting service is to host the directories of client companies. `example.com` actually hosts and partially manages the directories of two medium-sized companies,

HostedCompany1 and HostedCompany2. It also provides Internet access to a number of individual subscribers.

These are the access control rules that `example.com` wants to put in place:

- Grant anonymous access for read, search, and compare to the entire `example.com` tree for `example.com` employees ([Section 9.1, “Granting Anonymous Access”](#)).
- Grant write access to `example.com` employees for personal information, such as `homePhone` and `homePostalAddress` ([Section 9.2, “Granting Write Access to Personal Entries”](#)).
- Grant `example.com` employees the right to add any role to their entry, except certain critical roles ([Section 9.3, “Restricting Access to Key Roles”](#)).
- Grant the `example.com` Human Resources group all rights on the entries in the `People` branch ([Section 9.4, “Granting a Group Full Access to a Suffix”](#)).
- Grant all `example.com` employees the right to create group entries under the `Social Committee` branch of the directory and to delete group entries that they own ([Section 9.5, “Granting Rights to Add and Delete Group Entries”](#)).
- Grant all `example.com` employees the right to add themselves to group entries under the `Social Committee` branch of the directory ([Section 9.9, “Allowing Users to Add or Remove Themselves from a Group”](#)).
- Grant access to the directory administrator (role) of HostedCompany1 and HostedCompany2 on their respective branches of the directory tree, with certain conditions such as SSL authentication, time and date restrictions, and specified location ([Section 9.6, “Granting Conditional Access to a Group or Role”](#)).
- Deny individual subscribers access to the billing information in their own entries ([Section 9.7, “Denying Access”](#)).
- Grant anonymous access to the world to the individual subscribers subtree, except for subscribers who have specifically requested to be unlisted. (This part of the directory could be a consumer server outside of the firewall and be updated once a day.) See [Section 9.1, “Granting Anonymous Access”](#) and [Section 9.8, “Setting a Target Using Filtering”](#).

9.1. Granting Anonymous Access

Most directories are run such that you can anonymously access at least one suffix for read, search, or compare. For example, you might want to set these permissions if you are running a corporate personnel directory that you want employees to be able to search, such as a phonebook. This is the case at `example.com` internally and is illustrated in [Section 9.1.1, “ACI “Anonymous example.com””](#).

As an ISP, `example.com` also wants to advertise the contact information of all of its subscribers by creating a public phonebook accessible to the world. This is illustrated in [Section 9.1.2, “ACI “Anonymous World””](#).

9.1.1. ACI "Anonymous example.com"

In LDIF, to grant read, search, and compare permissions to the entire `example.com` tree to `example.com` employees, write the following statement:

```
aci: (targetattr != "userPassword")(version 3.0; acl "Anonymous
Example"; allow (read, search, compare) userdn= "ldap:///anyone"
and dns="*.example.com";)
```

This example assumes that the `aci` attribute is added to the `dc=example,dc=com` entry. The `userPassword` attribute is excluded from the scope of the ACI.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab in the **ACI name** field, type `Anonymous example.com`. Check that **All Users** opens in the list of users granted access permission.
4. In the **Rights** tab, select the checkboxes for `read`, `compare`, and `search` rights. Make sure the other checkboxes are clear.
5. In the **Targets** tab, click **This Entry** to display the `dc=example,dc=com` suffix in the **Target directory entry** field. In the attribute table, locate the `userPassword` attribute, and clear the corresponding checkbox.

All other checkboxes should be selected. This task is made easier if you click the **Name** header to organize the list of attributes alphabetically.

6. In the **Hosts** tab, click **Add**, and in the **DNS host filter** field, type `*.example.com`. Click **OK** to dismiss the dialog box.
7. Click **OK** in the **Access Control Editor** window.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.1.2. ACI "Anonymous World"

In LDIF, to grant read and search access of the individual subscribers subtree to the world, while denying access to information on unlisted subscribers, write the following statement:

```
aci: (targetfilter= "(! (unlistedSubscriber=yes))")
(targetattr="homePostalAddress || homePhone || mail") (version
3.0; acl "Anonymous World"; allow (read, search)
userdn="ldap:///anyone";)
```

This example assumes that the ACI is added to the `ou=subscribers,dc=example,dc=com` entry. It also assumes that every subscriber entry has an `unlistedSubscriber` attribute which is set to `yes` or `no`. The target definition filters out the unlisted subscribers based on the value of this attribute. For details on the filter definition, see [Section 9.8, “Setting a Target Using Filtering”](#).

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `Subscribers` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Anonymous World`. Check that `All Users` opens in the list of users granted access permission.
4. In the **Rights** tab, select the checkboxes for `read` and `search` rights. Make sure the other checkboxes are clear.
5. In the **Targets** tab, click **This Entry** to display the `ou=subscribers, dc=example,dc=com` suffix in the **Target directory entry** field.
6. In the **Filter for subentries** field, type the following filter:

```
(!(unlistedSubscriber=yes))
```

7. In the attribute table, select the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

All other checkboxes should be clear; if it is easier, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

8. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.2. Granting Write Access to Personal Entries

Many directory administrators want to allow internal users to change some but not all of the attributes in their own entry. The directory administrators at `example.com` want to allow users to change their own password, home telephone number, and home address, but nothing else. This is illustrated in [Section 9.2.1, “ACI “Write example.com””](#).

It is also `example.com`'s policy to let their subscribers update their own personal information in the `example.com` tree, provided that they establish an SSL connection to the directory. This is

illustrated in [Section 9.2.2, "ACI "Write Subscribers"'"](#).

9.2.1. ACI "Write example.com"



NOTE

By setting this permission, you are also granting users the right to delete attribute values.

Granting `example.com` employees the right to update their password, home telephone number, and home address has the following statement in LDIF:

```
aci: (targetattr="userPassword || homePhone ||
      homePostalAddress") (version 3.0; acl "Write example.com"; allow
      (write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `example-people` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Write example.com`. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area to `Special Rights`, and select `Self` from the search results list.
 - c. Click the **Add** button to list `Self` in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for `write` right. Make sure the other checkboxes are clear.
 5. In the **Targets** tab, click **This Entry** to display the `ou=example-people,dc=example,dc=com` suffix in the **Target directory entry** field. In the attribute table, select the checkboxes for the `homePhone`, `homePostalAddress`, and `userPassword` attributes.

All other checkboxes should be clear; if it is easier, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type `*.example.com`. Click **OK** to dismiss the dialog box.
7. Click **OK** in the **Access Control Editor** window.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.2.2. ACI "Write Subscribers"



NOTE

By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant `example.com` subscribers the right to update their password and home telephone number, write the following statement:

```
aci: (targetattr="userPassword || homePhone") (version 3.0; acl
  "Write Subscribers"; allow (write) userdn= "ldap://self" and
  authmethod="ssl");
```

This example assumes that the `aci` is added to the `ou=subscribers, dc=example, dc=com` entry.

`example.com` subscribers do not have write access to their home address because they might delete the attribute, and `example.com` needs that information for billing. Therefore, the home address is business-critical information.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the Subscribers entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Write Subscribers`. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area to `Special Rights`, and select `Self` from the search results list.
 - c. Click the **Add** button to list `Self` in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for `write`. Make sure the other checkboxes are clear.
 5. In the **Targets** tab, click **This Entry** to display the `ou=subscribers, dc=example,dc=com` suffix in the **Target directory entry** field.
 - a. In the **Filter for subentries** field, type the following filter:

```
(!(unlistedSubscriber=yes))
```

- b. In the attribute table, select the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

All other checkboxes should be clear; if necessary, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

- c. Optionally, to require users to authenticate using SSL, switch to manual editing by clicking the **Edit Manually** button, and add `authmethod=ssl` to the LDIF statement so that it reads as follows:

```
(targetattr="homePostalAddress || homePhone || mail")
(version 3.0; acl "Write Subscribers"; allow (write)
(userdn= "ldap:///self") and authmethod="ssl";)
```

6. Click **OK**.

The new ACL is added to the ones listed in the **Access Control Manager** window.

9.3. Restricting Access to Key Roles

You can use role definitions in the directory to identify functions that are critical to your business, the administration of your network and directory, or another purpose.

For example, you might create a `superAdmin` role by identifying a subset of your system administrators that are available at a particular time of day and day of the week at corporate sites worldwide, or you might want to create a `First Aid` role that includes all members of staff on a particular site that have done first aid training. For information on creating role definitions, see [Section 1, “Using Roles”](#).

When a role gives any sort of privileged user rights over critical corporate or business functions, consider restricting access to that role. For example, at `example.com`, employees can add any role to their own entry except the `superAdmin` role. This is illustrated in [Section 9.3.1, “ACI “Roles””](#).

9.3.1. ACI “Roles”

In LDIF, to grant `example.com` employees the right to add any role to their own entry except the `superAdmin` role, write the following statement:

```
aci: (targetattr = "nsroledn")
      (targetattrfilters="add=nsroledn:(nsroledn !=
      "cn=superAdmin,dc=example,dc=com")") (version 3.0; acl "Roles";
      allow (write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `example-people` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Roles`. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.
 - b. Set the **Search** area in the **Add Users and Groups** dialog box to `Special Rights`, and select `Self` from the search results list.
 - c. Click the **Add** button to list `Self` in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for `write`. Make sure the other checkboxes are clear.
5. In the **Targets** tab, click **This Entry** to use the `ou=example-people,dc=example,dc=com` suffix in the **Target directory entry** field.
6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type `*.example.com`. Click **OK** to dismiss the dialog box.
7. To create the value-based filter for roles, switch to manual editing by clicking the **Edit**

Manually button. Add the following to the beginning of the LDIF statement:

```
(targetattrfilters="add=nsroledn:(nsroledn != "cn=superAdmin,
dc=example,dc=com")")
```

The LDIF statement should read as follows:

```
(targetattrfilters="add=nsroledn:(nsroledn != "cn=superAdmin,
dc=example,dc=com")") (targetattr = "*") (target = "ldap:///
ou=example-people,dc=example,dc=com") (version 3.0; acl "Roles";
allow (write) (userdn = "ldap:///self") and (dns="*.example.com");)
```

8. Click **OK**.

The new ACL is added to the ones listed in the **Access Control Manager** window.

9.4. Granting a Group Full Access to a Suffix

Most directories have a group that is used to identify certain corporate functions. These groups can be given full access to all or part of the directory. By applying the access rights to the group, you can avoid setting the access rights for each member individually. Instead, you grant users these access rights simply by adding them to the group.

For example, when the Directory Server is set up with a typical process, an administrators group with full access to the directory is created by default.

At `example.com`, the `Human Resources` group is allowed full access to the `ou=example-people` branch of the directory so that they can update the employee database. This is illustrated in [Section 9.4.1, "ACI "HR"'"](#).

9.4.1. ACI "HR"

In LDIF, to grant the HR group all rights on the employee branch of the directory, use the following statement:

```
aci: (version 3.0; acl "HR"; allow (all) userdn=
"ldap:///cn=HRgroup,ou=example-people,dc=example,dc=com";)
```

This example assumes that the ACL is added to the `ou=example-people,dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `example-people` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to

display the **Access Control Manager**.

2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `HR`. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area to `Users and Groups`, and type `HRgroup` in the **Search for** field.

This example assumes that you have created an HR group or role. For more information on groups and roles, see [Chapter 5, Managing Entries with Roles, Class of Service, and Views](#).

- c. Click the **Add** button to list the HR group in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
 4. In the **Rights** tab, click the **Check All** button.
- All checkboxes are selected, except for proxy rights.

5. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.5. Granting Rights to Add and Delete Group Entries

Some organizations want to allow employees to create entries in the tree if it can increase their efficiency or if it can contribute to the corporate dynamics.

At `example.com`, there is an active social committee that is organized into various clubs, such as tennis, swimming, and skiing. Any `example.com` employee can create a group entry representing a new club. This is illustrated in [Section 9.5.1, "ACI "Create Group"'](#). Any `example.com` employee can become a member of one of these groups. This is illustrated in [Section 9.9.1, "ACI "Group Members"'](#) under [Section 9.9, "Allowing Users to Add or Remove Themselves from a Group"](#). Only the group owner can modify or delete a group entry. This is illustrated in [Section 9.5.2, "ACI "Delete Group"'](#).

9.5.1. ACI "Create Group"

In LDIF, to grant `example.com` employees the right to create a group entry under the `ou=Social Committee` branch, write the following statement:

```
aci: (target="ldap:///ou=social committee,dc=example,dc=com)
      (targetattrfilters="add=objectClass:(objectClass=groupOfNames)")
      (version 3.0; acl "Create Group"; allow (add))
```

```
(userdn= "ldap:///uid=*,ou=example-people,dc=example,dc=com")  
and dns="*.example.com";)
```



NOTE

This ACI does not grant write permission, which means that the entry creator cannot modify the entry.

This example assumes that the ACI is added to the `ou=social committee, dc=example, dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `Social Committee` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Create Group`. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.
 - b. Set the **Search** area to `Special Rights`, and select **All Authenticated Users** from the search results list.
 - c. Click the **Add** button to list **All Authenticated Users** in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for `add`. Make sure the other checkboxes are clear.
5. In the **Targets** tab, click **This Entry** to display the `ou=social committee, dc=example, dc=com` suffix in the **Target directory entry** field.
6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type `*.example.com`. Click **OK** to dismiss the dialog box.
7. To create the value-based filter that allows employees to add only group entries to this subtree, click the **Edit Manually** button. Add the following to the beginning of the LDIF statement:

```
(targetattrfilters="add=objectClass:(objectClass=groupOfNames) ")
```

The LDIF statement should read as follows:

```
(targetattrfilters="add=objectClass:(objectClass=groupOfNames) ")
(targetattr = "*") (target="ldap:///ou=social
committee,dc=example,dc=com)
(version 3.0; acl "Create Group"; allow (read,search,add)
(userdn= "ldap:///all") and (dns="*.example.com"); )
```

8. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.5.2. ACI "Delete Group"

In LDIF, to grant `example.com` employees the right to modify or delete a group entry which they own under the `ou=Social Committee` branch, write the following statement:

```
aci: (target="ou=social committee,dc=example,dc=com)
(targetattrfilters="del=objectClass:(objectClass=groupOfNames) ")
(version 3.0; acl "Delete Group"; allow (delete) userattr=
"owner#GROUPDN";)
```

This example assumes that the `aci` is added to the `ou=social committee, dc=example,dc=com` entry.



NOTE

Using the Console is not an effective way of creating this ACI because it requires manually editing the ACI to create the target filter and to check group ownership.

9.6. Granting Conditional Access to a Group or Role

In many cases, when you grant a group or role privileged access to the directory, you want to ensure that those privileges are protected from intruders trying to impersonate your privileged users. Therefore, in many cases, access control rules that grant critical access to a group or role are often associated with a number of conditions.

`example.com` has created a directory administrator role for each of its hosted companies, `HostedCompany1` and `HostedCompany2`. It wants these companies to be able to manage their own data and implement their own access control rules while securing it against intruders. For this reason, `HostedCompany1` and `HostedCompany2` have full rights on their respective branches

of the directory tree, provided the following conditions are fulfilled:

- Connection authenticated using SSL
- Access requested between 8 a.m. and 6 p.m., Monday through Thursday
- Access requested from a specified IP address for each company

These conditions are illustrated in a single ACI for each company, `HostedCompany1` and `HostedCompany2`. Because the content of these ACIs is the same, the examples below illustrate the `HostedCompany1` ACI only.

9.6.1. ACI "HostedCompany1"

In LDIF, to grant `HostedCompany1` full access to their own branch of the directory under the conditions stated above, write the following statement:

```
aci:(target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com")
  (targetattr= "*" ) (version 3.0; acl "HostedCompany1";allow (all)
  (roledn="ldap:///cn=DirectoryAdmin,ou=HostedCompany1,
  ou=corporate-clients, dc=example,dc=com") and
  (authmethod="ssl") and (dayofweek="Mon,Tues,Wed,Thu") and (timeofday >=
  "0800" and
  timeofday <= "1800") and (ip="255.255.123.234"); )
```

This example assumes that the ACI is added to the `ou=HostedCompany1, ou=corporate-clients,dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `HostedCompany1` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, type `HostedCompany1` in the **ACI name** field. In the list of users granted access permission, do the following:
 - a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area to `Users and Groups`, and type `DirectoryAdmin` in the **Search For** field.

This example assumes that you have created an administrators role with a `cn` of `DirectoryAdmin`.

- c. Click the **Add** button to list the administrators role in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, click the **Check All** button.
 5. In the **Targets** tab, click **This Entry** to display the
`ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com` suffix in the **Target directory entry** field.
 6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **IP address host filter** field, type `255.255.123.234`. Click **OK**.

The IP address must be a valid IP address for the host machine that the `HostedCompany1` administrators use to connect to the `example.com` directory.

7. In the **Times** tab, select the block time corresponding to Monday through Thursday and 8 a.m. to 6 p.m.

A message appears below the table that specifies the selected time block.

8. To enforce SSL authentication from `HostedCompany1` administrators, switch to manual editing by clicking the **Edit Manually** button. Add the following to the end of the LDIF statement:

```
and (authmethod="ssl")
```

The LDIF statement should be similar to the following:

```
aci: (targetattr = "*")
(target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com")
(version 3.0; acl "HostedCompany1"; allow (all) (roledn=
"ldap:///cn=DirectoryAdmin,ou=HostedCompany1,ou=corporate-clients,
dc=example,dc=com") and
(dayofweek="Mon,Tues,Wed,Thu") and (timeofday >= "0800" and timeofday
<= "1800") and
(ip="255.255.123.234") and (authmethod="ssl"); )
```

9. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.7. Denying Access

If your directory holds business-critical information, it may be necessary to specifically deny access to it.

For example, `example.com` wants all subscribers to be able to read billing information such as connection time or account balance under their own entries but explicitly wants to deny write access to that information. This is illustrated in [Section 9.7.1, “ACI “Billing Info Read””](#) and [Section 9.7.2, “ACI “Billing Info Deny””](#), respectively.

9.7.1. ACI “Billing Info Read”

In LDIF, to grant subscribers permission to read billing information in their own entry, write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
  3.0; acl "Billing Info Read"; allow (search,read) userdn=
  "ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the `ou=subscribers,dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `Subscribers` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Billing Info Read`. In the list of users granted access permission, do the following:

- a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area in the **Add Users and Groups** dialog box to `Special Rights`, and select `Self` from the search results list.
 - c. Click the **Add** button to list `Self` in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkboxes for `search` and `read` rights. Make sure the other checkboxes are clear.
 5. In the **Targets** tab, click **This Entry** to display the `ou=subscribers, dc=example,dc=com` suffix in the **Target directory entry** field. In the attribute table, select the checkboxes for the `connectionTime` and `accountBalance` attributes.

All other checkboxes should be clear; if it is made easier, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize

them alphabetically, and select the appropriate ones.

This example assumes that you have added the *connectionTime* and *accountBalance* attributes to the schema.

6. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.7.2. ACI "Billing Info Deny"

In LDIF, to deny subscribers permission to modify billing information in their own entry, write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
3.0; acl "Billing Info Deny"; deny (write) userdn="ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the *ou=subscribers,dc=example,dc=com* entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the *Subscribers* entry under the *example.com* node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type *Billing Info Deny*. In the list of users granted access permission, do the following:
 - a. Select and remove *All Users*, then click **Add**.

The **Add Users and Groups** dialog box opens.
 - b. Set the **Search** area in the **Add Users and Groups** dialog box to *Special Rights*, and select *Self* from the search results list.
 - c. Click the **Add** button to list *Self* in the list of users who are granted access permission.
 - d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for *write*. Make sure the other checkboxes are clear.
5. Click the **Edit Manually** button, and, in the LDIF statement that opens, change the word *allow* to *deny*.
6. In the **Targets** tab, click **This Entry** to display the *ou=subscribers, dc=example,dc=com* suffix in the **Target directory entry** field. In the attribute table, select the checkboxes for the

connectionTime and *accountBalance* attributes.

All other checkboxes should be clear; if it is easier, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

This example assumes that the *connectionTime* and *accountBalance* attributes were added to the schema.

7. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.8. Setting a Target Using Filtering

To set access controls that allow access to a number of entries that are spread across the directory, consider using a filter to set the target.



NOTE

Because search filters do not directly name the object for which you are managing access, it is easy to allow or deny access to the wrong objects unintentionally, especially as your directory becomes more complex. Additionally, filters can make it difficult to troubleshoot access control problems within your directory.

For example, the following ACI grants user *bjensen* write access to the department number, home phone number, home postal address, and manager attributes for all members of the accounting organization.

```
aci: (targetattr="departmentNumber || homePhone || homePostalAddress ||
manager")
(targetfilter="(uid=bjensen)") (version 3.0; acl "Filtered ACL"; allow
(write)
userdn ="ldap:///cn=*,ou=accounting, dc=example,dc=com";)
```

Before you can set these permissions, you must create the accounting branch point (*ou=accounting,dc=example,dc=com*). You can create organizational unit branch points in the **Directory** tab on the Directory Server Console.

9.9. Allowing Users to Add or Remove Themselves from a Group

Many directories set ACIs that allow users to add or remove themselves from groups. This is useful, for example, for allowing users to add and remove themselves from mailing lists.

At `example.com`, employees can add themselves to any group entry under the `ou=social committee` subtree. This is illustrated in [Section 9.9.1, “ACI “Group Members””](#).

9.9.1. ACI “Group Members”

In LDIF, to grant `example.com` employees the right to add or delete themselves from a group, write the following statement:

```
aci: (targetattr="member")(version 3.0; acl "Group Members"; allow
(selfwrite)
(userdn= "ldap:///uid=*,ou=example-people,dc=example,dc=com" ) ;)
```

This example assumes that the ACI is added to the `ou=social committee, dc=example,dc=com` entry.

From the Console, set this permission by doing the following:

1. In the **Directory** tab, right-click the `example-people` entry under the `example.com` node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type `Group Members`. In the list of users granted access permission, do the following:

- a. Select and remove `All Users`, then click **Add**.

The **Add Users and Groups** dialog box opens.

- b. Set the **Search** area in the **Add Users and Groups** dialog box to `Special Rights`, and select **All Authenticated Users** from the search results list.
- c. Click the **Add** button to list **All Authenticated Users** in the list of users who are granted access permission.
- d. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the checkbox for `selfwrite`. Make sure the other checkboxes are clear.
5. In the **Targets** tab, type `dc=example,dc=com` suffix in the **Target directory entry** field. In the attribute table, select the checkbox for the `member` attribute.

All other checkboxes should be clear; if it is easier, click the **Check None** button to clear the checkboxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

6. Click **OK**.

The new ACI is added to the ones listed in the **Access Control Manager** window.

9.10. Defining Permissions for DNs That Contain a Comma

DNs that contain commas require special treatment within your LDIF ACI statements. In the target and bind rule portions of the ACI statement, commas must be escaped by a single backslash (\). For example:

```
dn: dc=example.com Bolivia\, S.A.,dc=com
objectClass: top
objectClass: organization
aci: (target="ldap:///dc=example.com Bolivia\,S.A.,dc=com")(targetattr=*)
      (version 3.0; acl "aci 2"; allow (all)
        groupdn = "ldap:///cn=Directory Administrators,dc=example.com Bolivia\,
        S.A.,dc=com";)
```

9.11. Proxied Authorization ACI Example

Proxied authorization allows one user to bind and perform operation as another user. For example, example.com has an accounting program which must be able to bind to the directory as an accounting administrator in order to write data. This authorization assumes three things:

- The client application's bind DN is "uid=MoneyWizAcctSoftware, ou=Applications,dc=example,dc=com".
- The targeted subtree to which the client application is requesting access is ou=Accounting,dc=example,dc=com.
- An accounting administrator with access permissions to the ou=Accounting,dc=example,dc=com subtree exists in the directory.

In order for the client application to gain access to the accounting subtree, using the same access permissions as the accounting administrator, two ACIs must be set:

- The accounting administrator must have access permissions to the ou=Accounting,dc=example,dc=com subtree, so the following ACI grants all rights to the accounting administrator entry:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com") (targetattr="*")
      (version 3.0; acl "allowAll-AcctAdmin"; allow (all)
        userdn="ldap://uid=AcctAdministrator,ou=Administrators,dc=example,dc=com")
```

- There must be an ACI granting proxy rights to the client application in the directory:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com") (targetattr="*")
      (version 3.0; acl "allow proxy-accounting software"; allow (proxy)
      userdn="ldap://uid=MoneyWizAcctSoftware,ou=Applications,dc=example,dc=com")
```

With this ACL in place, the **MoneyWizAcctSoftware** client application can bind to the directory and send an LDAP command such as `ldapsearch` or `ldapmodify` that requires the access rights of the proxy DN.

If the client performs an `ldapsearch` command, the command must include the following controls:

```
ldapmodify -D "uid=MoneyWizAcctSoftware,ou=Applications,dc=example,dc=com"
-w secretpwd
-y "uid=AcctAdministrator,ou=Administrators,dc=example,dc=com"
```

The client or application (**MoneyWizAcctSoftware**) binds as itself but is granted the privileges of the proxy entry (`AcctAdministrator`). The client does not need the password of the proxy entry.



NOTE

There are some restrictions on binding with proxy authorization. You cannot use the Directory Manager's DN (root DN) as a proxy DN. Additionally, if Directory Server receives more than one proxied authentication control, an error is returned to the client application, and the bind attempt is unsuccessful.

10. Advanced Access Control: Using Macro ACLs

In organizations that use repeating directory tree structures, it is possible to optimize the number of ACLs used in the directory by using macros. Reducing the number of ACLs in your directory tree makes it easier to manage your access control policy and improves the efficiency of ACL memory usage.

Macros are placeholders that are used to represent a DN, or a portion of a DN, in an ACL. You can use a macro to represent a DN in the target portion of the ACL or in the bind rule portion, or both. In practice, when Directory Server gets an incoming LDAP operation, the ACL 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 ACL normally.

10.1. Macro ACL Example

Figure 6.3, "Example Directory Tree for Macro ACLs" 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.com` directory tree stores the suffixes

dc=hostedCompany2, dc=example,dc=com **and** dc=hostedCompany3,dc=example,dc=com.

The ACIs that apply in the directory tree also have a repeating pattern. For example, the following ACI is located on the `dc=hostedCompany1,dc=example,dc=com` node:

```
aci: (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
      (version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com";)
```

This ACI grants read and search rights to the `DomainAdmins` group to any entry in the `dc=hostedCompany1,dc=example,dc=com` tree.

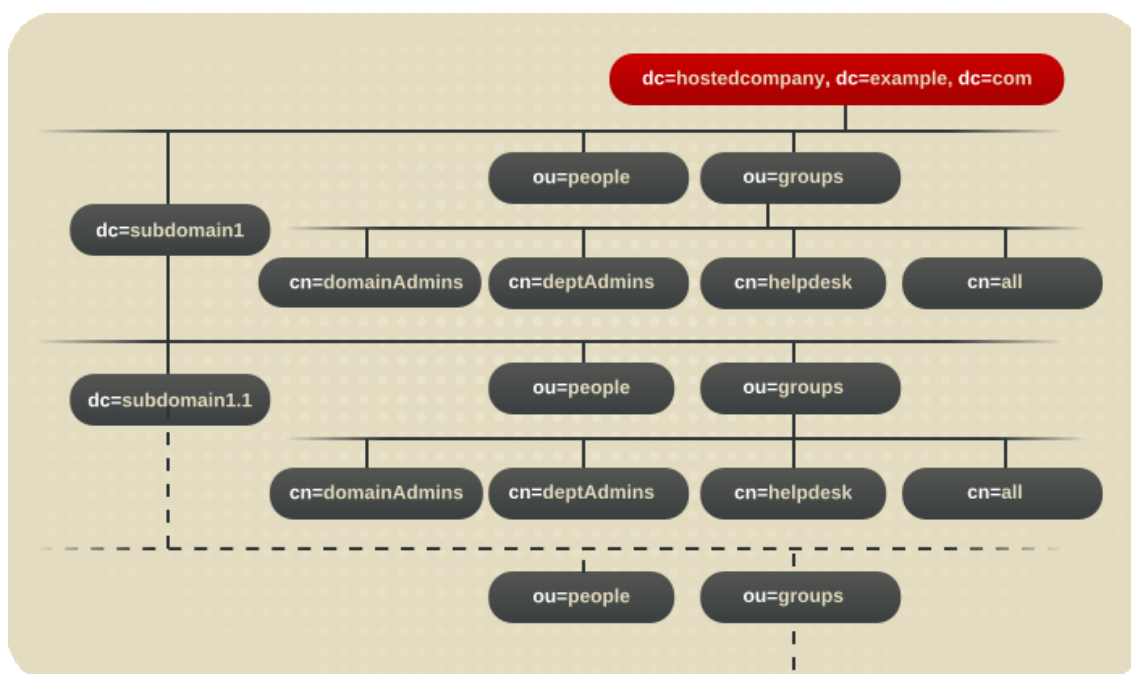


Figure 6.3. Example Directory Tree for Macro ACIs

The following ACI is located on the `dc=hostedCompany1,dc=example,dc=com` node:

```
aci: (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
      (version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany1,dc=example,dc=com` node:

```
aci: (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
```



```
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com"
```

The following ACI is located on the `dc=hostedCompany2,dc=example,dc=com` node:

```
aci: (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
      (version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany2,dc=example,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany2,dc=example,dc=com` node:

```
aci: (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
      (version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,
dc=subdomain1,dc=hostedCompany2,dc=example,dc=com";)
```

In the four ACIs shown above, the only differentiator is the DN specified in the `groupdn` keyword. By using a macro for the DN, it is possible to replace these ACIs by a single ACI at the root of the tree, on the `dc=example,dc=com` node. This ACI reads as follows:

```
aci: (target="ldap:///ou=Groups,($dn),dc=example,dc=com")
      (targetattr="*)(targetfilter=(objectClass=nsManagedDomain))
      (version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=com";)
```

The `target` keyword, which was not previously used, 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.

10.2. 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 6.9, “Macros in ACI Keywords”](#) shows in what parts of the ACI you can use DN macros:

Macro	ACI Keyword
<code>(\$dn)</code>	target, targetfilter, userdn, roledn, groupdn, userattr
<code>[\$dn]</code>	targetfilter, userdn, roledn, groupdn, userattr
<code>(\$attr.attrName)</code>	userdn, roledn, groupdn, userattr

Table 6.9. Macros in ACI Keywords

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.

10.2.1. 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.

10.2.2. 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,dc=example,dc=com"`. If the bind DN is a member of that group, the matching process stops, and the ACI is evaluated. If it does not match, the process continues.

3. [`$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 ACL:

```
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.

10.2.3. 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 ACL, 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 ACL according to the normal ACL 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"
```

11. Access Control and Replication

ACIs are stored as attributes of entries; therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated like any other attribute.

ACIs are always evaluated on the Directory Server that services the incoming LDAP requests. This means that when a consumer server receives an update request, it returns a referral to the supplier server before evaluating whether the request can be serviced on the supplier.

12. Compatibility with Earlier Releases

Some ACI keywords that were used in earlier releases of Directory Server have been deprecated. However, for reasons of backward compatibility, the following keywords are still supported:

- userdnattr
- groupdnattr

Therefore, if you have set up a replication agreement between a legacy supplier server and a version 8.0 consumer, there should not be any problems in the replication of ACIs.

Managing User Accounts and Passwords

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 DN.

1. Managing the Password Policy

A password policy minimizes the risks of using passwords by enforcing the following:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords.
- The password syntax must meet certain complexity requirements.

After establishing a password policy, which can be for the entire directory or for specific subtrees or users, 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.

This section provides information about configuring password and account lockout policies:

- [Section 1.1, “Configuring the Password Policy”](#)
- [Section 1.2, “Setting User Passwords”](#)
- [Section 1.3, “Password Change Extended Operation”](#)
- [Section 1.4, “Configuring the Account Lockout Policy”](#)
- [Section 1.5, “Managing the Password Policy in a Replicated Environment”](#)
- [Section 1.6, “Synchronizing Passwords”](#)

1.1. Configuring the Password Policy

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).

Essentially, the password policy is comprised of the following information:

- *The type or level of password policy checks.* This information indicates whether the server should check for and enforce a global password policy or local (subtree/user level) password policies.
- *Password add and modify information.* The password information includes password syntax and password history details.
- *Bind information.* The bind information includes the number of grace logins permitted, password aging attributes, and tracking bind failures.

The sections that follow describe the procedures for configuring the password policy:

- [Section 1.1.1, “Configuring a Global Password Policy Using the Console”](#)
- [Section 1.1.2, “Configuring a Subtree/User Password Policy Using the Console”](#)
- [Section 1.1.3, “Configuring a Global Password Policy Using the Command-Line”](#)
- [Section 1.1.4, “Configuring Subtree/User Password Policy Using the Command-Line”](#)



NOTE

After configuring the password policy, we recommend configuring an account lockout policy. For details, see [Section 1.4, “Configuring the Account Lockout Policy”](#).

1.1.1. Configuring a Global Password Policy Using the Console

To set up or modify the password policy for an entire directory, do the following:

1. In the Directory Server Console, select the **Configuration** tab and then the **Data** node.
2. In the right pane, select the **Passwords** tab.

This tab contains the password policy for the entire Directory Server.

3. Check the **Enable fine-grained password policy** checkbox. Enabling the password policy makes the other sections on the screen active.
4. To require users to change their password the first time they log on, select the **User must change password after reset** checkbox. If this checkbox is selected, only the Directory Manager is authorized to reset the user's password. A regular administrative user cannot

force the users to update their password.

5. To allow users to change their own passwords, select the **User may change password** checkbox.
6. To prevent users from changing their password for a specific duration, enter the number of days in the **Allow changes in X day(s)** text box.
7. For the server to maintain a history list of passwords used by each user, select the **Keep password history** checkbox. Enter the number of passwords for the server to keep for each user in the **Remember X passwords** text box.
8. If user passwords should not expire, select the **Password never expires** radio button.
9. To require users to change their passwords periodically, select the **Password expires after X days** radio button, and then enter the number of days that a user password is valid.

The maximum value for the password age is derived by subtracting January 18, 2038, from today's date. The entered value must not be set to the maximum value or too close to the maximum value. Setting the value to the maximum value can cause the Directory Server to fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, correct the `passwordMaxAge` attribute value in the `dse.ldif` file.

A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to 8640000 seconds (100 days).

- 10 If the **Password expire after X days** radio button is selected, specify how long before the password expires to send a warning to the user. In the **Send Warning X Days Before Password Expires** text enter the number of days before password expiration to send a warning.



NOTE

It is not necessary to configure the Directory Server to send a warning to users. The Directory Server automatically issues a warning the next time the user attempts to log into the Directory Server Console that the password will soon expire or has expired. This is analogous to an operating system warning that reads "Warning: password will expire in 7 days" when a user logs in.

- 11 For the server to check the syntax of a user password to make sure it meets the minimum requirements set by the password policy, select the **Check Password Syntax** checkbox. Then, specify required password complexity, such as the minimum length and required number of numeric and special characters. The password syntax requirements are described more in [Table 7.1, "Password Policy Attributes"](#).

- 12 From the **Password Encryption** pull-down menu, select the encryption method for the

server to use when storing passwords.

For detailed information about the encryption methods, refer to the `passwordStorageScheme` attribute in [Table 7.1, “Password Policy Attributes”](#).

The **Password Encryption** menu might contain other encryption methods, as the directory dynamically creates the menu depending upon the existing encryption methods it finds in the directory.

13. Click **Save**.

1.1.2. Configuring a Subtree/User Password Policy Using the Console

1. Enable fine-grained password policy globally.
 - a. Select the **Configuration** tab, then click the **Data** node.
 - b. In the right pane, select the **Passwords** tab.
 - c. Check the **Enable fine-grained password policy** checkbox.
 - d. Click **Save**.



NOTE

The password policy *must* be enabled globally before it will be applied locally. No other global password policy features must be set, and the global password policy will not override the local policy if they differ.

2. Create the local password policy for the subtree or user.
 - a. Select the **Directory** tab.
 - b. In the navigation pane, select the subtree or user entry for which to set up the password policy.
 - c. From the **Object** menu, select the **Manage Password Policy** option, and then select the **For user** or **For subtree**.

Either the **User Password Policy** or **Subtree Password Policy** window appears.
 - d. In the **Passwords** tab, select the **Create subtree/user level password policy** checkbox to add the required attributes, fill in the appropriate values, and click **Save**.
 - e. In the **Account Lockout** tab, specify the appropriate information, and click **Save**.

1.1.3. Configuring a Global Password Policy Using the Command-Line

To set up the password policy for a subtree or user, add the required entries and attributes at the subtree or user level, set the appropriate values to the password policy attributes, and enable fine-grained password policy checking.

This section describes the attributes to create a password policy for the entire server (globally) using `ldapmodify` to change these attributes in the `cn=config` entry.

[Table 7.1, “Password Policy Attributes”](#) describes the attributes available to configure the password policy.

Attribute Name	Definition
<code>passwordGraceLimit</code>	This attribute indicates the number of grace logins permitted when a user's password is expired. When set to a positive number, the user will be allowed to bind with the expired password for that many times. For the global password policy, the attribute is defined under <code>cn=config</code> . By default, this attribute is set to 0, which means grace logins are not permitted.
<code>passwordMustChange</code>	When <code>on</code> , this attribute requires users to change their passwords when they first login to the directory or after the password is reset by the Directory Manager. The user is required to change their password even if user-defined passwords are disabled. If this attribute is set to <code>off</code> , passwords assigned by the Directory Manager should not follow any obvious convention and should be difficult to discover. This attribute is <code>off</code> by default.
<code>passwordChange</code>	When <code>on</code> , this attribute indicates that users may change their own password. Allowing users to set their own passwords runs the risk of users choosing passwords that are easy to remember. However, setting good passwords for the user requires a significant administrative effort. In addition, providing passwords to users that are not meaningful to them runs the risk that users will write the password down somewhere that can be discovered. This attribute is <code>on</code> by default.
<code>passwordExp</code>	When <code>on</code> , this attribute indicates that the

Attribute Name	Definition
	user's password will expire after an interval given by the <i>passwordMaxAge</i> attribute. Making passwords expire helps protect the directory data because the longer a password is in use, the more likely it is to be discovered. This attribute is <code>off</code> by default.
<code>passwordMaxAge</code>	This attribute indicates the number of seconds after which user passwords expire. To use this attribute, enable password expiration using the <i>passwordExp</i> attribute. This attribute is a dynamic parameter in that its maximum value is derived by subtracting January 18, 2038, from today's date. The attribute value must not be set to the maximum value or too close to the maximum value. If the value is set to the maximum value, Directory Server may fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, correct the <i>passwordMaxAge</i> attribute value in the <code>dse.ldif</code> file. A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to <code>8640000</code> seconds (100 days).
<code>passwordWarning</code>	This attribute indicates the number of seconds before a warning message is sent to users whose password is about to expire. Depending on the LDAP client application, users may be prompted to change their password when the warning is sent. Both Red Hat Directory Express and the Directory Server Gateway provide this functionality. By default, the directory sends the warning <code>86400</code> seconds (1 day) before the password is about to expire. However, a password never expires until the warning message has been set. Therefore, if users don't bind to the Directory Server for longer than the <i>passwordMaxAge</i> , they will still get the warning message in time to change their password.
<code>passwordMinAge</code>	This attribute indicates the number of seconds that must pass before a user can change their

Attribute Name	Definition
	<p>password. Use this attribute in conjunction with the <i>passwordInHistory</i> attribute to discourage users from reusing old passwords. For example, setting the minimum password age to 2 days prevents users from repeatedly changing their passwords during a single session to cycle through the password history and reuse an old password once it has been removed from the history list. The minimum age can be from 0 to 2147472000 seconds (24,855 days). A value of zero indicates that the user can change the password immediately. The default value of this attribute is 0.</p>
passwordHistory	<p>This attribute indicates whether the directory stores a password history. When set to <i>on</i>, the directory stores the number of passwords specified in the <i>passwordInHistory</i> attribute in a history. If a user attempts to reuse one of the passwords, the password will be rejected. When this attribute is set to <i>off</i>, any passwords stored in the history remain there. When this attribute is set back to <i>on</i>, users will not be able to reuse the passwords recorded in the history before the attribute was disabled. This attribute is <i>off</i> by default, meaning users can reuse old passwords.</p>
passwordInHistory	<p>This attribute indicates the number of passwords the directory stores in the history. There can be 2 to 24 passwords stored in the history. This feature is not enabled unless the <i>passwordHistory</i> attribute is set to <i>on</i>. This attribute is set to 6 by default.</p>
passwordCheckSyntax	<p>When <i>on</i>, this attribute indicates that the password syntax is checked by the server before the password is saved. Password syntax checking ensures that the password string meets or exceeds the length and complexity requirements and that the string does not contain any <i>trivial</i> words. A trivial word is any value stored in the <i>uid</i>, <i>cn</i>, <i>sn</i>, <i>givenName</i>, <i>ou</i>, or <i>mail</i> attributes of the user's entry. This attribute is <i>off</i> by default.</p>

Attribute Name	Definition
passwordMinLength	This attribute specifies the minimum number of characters that must be used in passwords. Shorter passwords are easier to crack. Passwords can be two (2) to 512 characters long. Generally, a length of eight characters is long enough to be difficult to crack but short enough for users to remember without writing it down. This attribute is set to 8 by default.
passwordMaxRepeats	This attribute set the maximum number of times that the same character can be used in row, such as <code>aaaaa</code> . Setting the attribute to 0 means that there is no limit on how many time a character can be repeated. This attribute is set to 0 by default.
passwordMinAlphas	This attribute sets the minimum number of alphabetic chracters that must be used in the password. This setting does not set any requirements for the letter case; requirements for the minimum number of lowercase and uppercase letters are set in the <code>passwordMinLower</code> and <code>passwordMinUpper</code> attributes, respectively. By default, this attribute is set to 0, meaning there is no required minimum.
passwordMinDigits	This attribute sets the minimum number of numeric characters (0 through 9) which must be used in the password. By default, this attribute is set to 0, meaning there is no required minimum.
passwordMinSpecials	This attribute sets the minimum number of special ASCII characters, such as <code>!@#\$%&*</code> , which must be used in the password. By default, this attribute is set to 0, meaning there is no required minimum.
passwordMinLowers	This attribute sets the minimum number of lower case alphabetic characters, a to z, which must be used in the password. By default, this attribute is set to 0, meaning there is no required minimum.
passwordMinCategories	This attribute sets the minimum number of categories which must be used in the password. There are eight categories available:

Attribute Name	Definition
	<p>Uppercase letters (A to Z)</p> <p>Lowercase letters (a to z)</p> <p>Numbers (0 through 9)</p> <p>Special ASCII characters, such as \$</p> <p>ASCII alphabetic characters, regardless of case (a to z and A to Z)</p> <p>8-bit characters</p> <p>Repeated characters, such as aaaaaa</p> <p>This attribute is set to 3 by default.</p>
passwordMinUppers	<p>This attribute sets the minimum number of upper case alphabetic characters, A to Z, which must be used in the password. By default, this attribute is set to 0, meaning there is no required minimum.</p>
passwordTokenLength	<p>This attribute sets the minimum length for any tokens used with Directory Server. The token length can be from 1 to 64 characters. This attribute is set to 3 by default.</p>
passwordMin8bit	<p>This attribute sets the minimum number of 8-bit chracters used in the password. The default number is 0, meaning none are required.</p>
passwordStorageScheme	<p>This attribute specifies the type of encryption used to store Directory Server passwords. The following encryption types are supported by Directory Server:</p> <p><i>SSHA (Salted Secure Hash Algorithm).</i> This method is recommended as it is the most secure. The Directory Server supports SSHA, SSHA-256, SSHA-384, and SSHA-512. SSHA is the default method.</p> <p><i>SHA (Secure Hash Algorithm).</i> A one-way hash algorithm; it is supported only for backwards compatibility with Directory Server 4.x and should not be used otherwise. This includes support for SHA, SHA-256, SHA-384, and SHA-512 algorithms, which protects against some insecurities in the SHA-1 algorithm.</p> <p><i>MD5.</i> MD5 is not as secure as SSHA but is available for legacy applications require it.</p> <p><i>crypt.</i> The UNIX crypt algorithm, provided for</p>

Attribute Name	Definition
	<p>compatibility with UNIX passwords.</p> <p><i>clear</i>. This encryption type indicates that the password will appear in plain text.</p> <p>The only password storage scheme that can be used with SASL DIGEST-MD5 is <code>CLEAR</code>.</p> <p>Passwords stored using <code>crypt</code>, <code>SHA</code>, or <code>SSHA</code> formats cannot be used for secure login through SASL Digest MD5. To provide a customized storage scheme, consult Red Hat professional services.</p>

Table 7.1. Password Policy Attributes

1.1.4. Configuring Subtree/User Password Policy Using the Command-Line

To configure a subtree or user level password policy, do the following:

1. Add the required attributes to the subtree or user entries by running the `ns-newpwpolicy.pl` script.

The command syntax for the script is as follows:

```
ns-newpwpolicy.pl [-D rootDN] { -w password | -w - | -j filename }[-p port]
[-h host]
    -U userDN -S suffixDN
```

For updating a subtree entry, use the `-s` option. For updating a user entry, use the `-U` option. The `ns-newpwpolicy.pl` script accepts only one user or subtree entry at a time. It can, however, accept both user and suffix entries at the same time. For details about the script, see the *Directory Server Configuration, Command, and File Reference*.

2. The script adds the required attributes depending on whether the target entry is a subtree or user entry.

For a subtree (for example, `ou=people, dc=example, dc=com`), the following entries are added:

- A container entry (`nsPwPolicyContainer`) at the subtree level for holding various password policy-related entries for the subtree and all its children. For example:

```
dn: cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectClass: top
objectClass: nsContainer
```



```
cn: nsPwPolicyContainer
```

- The actual password policy specification entry (*nsPwPolicyEntry*) for holding all the password policy attributes that are specific to the subtree. For example:

```
dn: cn="cn=nsPwPolicyEntry,ou=people,dc=example,dc=com",  
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com  
objectclass: top  
objectclass: extensibleObject  
objectclass: ldapsubentry  
objectclass: passwordpolicy
```

- The CoS template entry (*nsPwTemplateEntry*) that has the *pwdpolicysubentry* value pointing to the above (*nsPwPolicyEntry*) entry. For example:

```
dn: cn="cn=nsPwTemplateEntry,ou=people,dc=example,dc=com",  
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com  
objectclass: top  
objectclass: extensibleObject  
objectclass: costemplate  
objectclass: ldapsubentry  
cosPriority: 1  
pwdpolicysubentry: cn="cn=nsPwPolicyEntry,ou=people,dc=example,dc=com",  
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
```

- The CoS specification entry at the subtree level. For example:

```
dn: cn=nsPwPolicy_cos,ou=people,dc=example,dc=com  
objectclass: top  
objectclass: LDAPsubentry  
objectclass: cosSuperDefinition  
objectclass: cosPointerDefinition  
cosTemplateDn: cn="cn=nsPwTemplateEntry,ou=people,dc=example,dc=com",  
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com  
cosAttribute: pwdpolicysubentry default operational
```

For a user (for example, *uid=jdoe*, *ou=people*, *dc=example*, *dc=com*), the following entries are added:

- A container entry (*nsPwPolicyContainer*) at the parent level for holding various password policy related entries for the user and all its children. For example:

```
dn: cn=nsPwPolicyContainer, ou=people, dc=example, dc=com  
objectClass: top  
objectClass: nsContainer  
cn: nsPwPolicyContainer
```

- The actual password policy specification entry (*nsPwPolicyEntry*) for holding the password policy attributes that are specific to the user. For example:

```
dn: cn="cn=nsPwPolicyEntry,uid=jdoe,ou=people,dc=example,dc=com",
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

3. Assign the value of the above entry DN to the *pwdpolicysubentry* attribute of the target entry. For example, this assigns the password policy to the user entry:

```
dn: uid=jdoe,ou=people,dc=example,dc=com
changetype: modify
replace: pwdpolicysubentry
pwdpolicysubentry:
"cn=nsPwPolicyEntry,uid=jdoe,ou=people,dc=example,dc=com",
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
```

4. Set the password policy attributes of subtree or user entry with the appropriate values.

[Table 7.1, “Password Policy Attributes”](#) describes the attributes available to configure the password policy. The `ldapmodify` utility can be used to change these attributes in the `cn=config` entry.



NOTE

The `nsslapd-pwpolicy-local` attribute of the `cn=config` entry controls the type of password policy the server enforces. By default, this attribute is disabled (`off`). When the attribute is disabled, the server only checks for and enforces the global password policy; the subtree and user level password policies are ignored. When the `ns-newpwpolicy.pl` script runs, it first checks for the specified subtree and user entries and, if they exist, modifies them. After updating the entries successfully, the script sets the `nsslapd-pwpolicy-local` configuration parameter to `on`. If the subtree and user level password policy should not be enabled, be sure to set `nsslapd-pwpolicy-local` to `off` after running the script.

To turn off user and subtree level password policy checks, set the `nsslapd-pwpolicy-local` attribute to `off` by modifying the `cn=config` entry. For example: ¹

```
ldapmodify -h myserver -p 389 -D "cn=directory manager" -w secretpwd

dn: cn=config
```

```
changetype: modify
replace: nsslapd-pwpolicy-local: on
nsslapd-pwpolicy-local: off
```

This attribute can also be disabled by modifying it directly in the configuration file (`dse.ldif`).

1. Stop the server.²

```
service dirsrv stop instance
```

2. Open the `dse.ldif` file in a text editor.
3. Set the value of `nsslapd-pwpolicy-local` to `off`, and save.

```
nsslapd-pwpolicy-local: off
```

4. Start the server.

```
service dirsrv start instance
```

1.2. Setting User Passwords

An entry can be used to bind to the directory only if it has a `userpassword` attribute and if it has not been inactivated. Because user passwords are stored in the directory, the user passwords can be set or reset with any LDAP operation, like `ldapmodify`.¹

For information on creating and modifying directory entries, see [Chapter 2, Creating Directory Entries](#). For information on inactivating user accounts, refer to [Section 2, “Inactivating Users and Roles”](#).

Passwords can also be set and reset in the **Users and Groups** area of the Red Hat Administration Server or the Directory Server Gateway. For information on how to use the **Users and Groups** area, see the online help that is available in the Red Hat Administration Server. For information on how to use the Gateway to create or modify directory entries, see the Gateway online help.

1.3. Password Change Extended Operation


¹ The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

² The commands to stop and start the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, “Starting and Stopping Servers”](#).

While most passwords can be changed through the Console and other Directory Server features or through the `ldapmodify` operation, there are some passwords that cannot be changed through regular LDAP operations. These passwords may be stored outside the Directory Server, such as passwords stored in a SASL application. These passwords can be modified through the *password change extended operation*.

Directory Server supports the password change extended operation as defined in RFC 3062, so users can change their passwords, using a suitable client, in a standards-compliant way. Directory Server does not include a client application for the password change extended operation. However, the `ldappasswd` utility can be used as follows:

```
ldappasswd -h hostname -p secure_port -Z -P /path/to/cert8.db -D bindDN -w  
bindPassword  
[-a oldPassword] -s newPassworduser
```

Parameter	Description
-h	Gives the hostname of the Directory Server.
-p	Gives the port number of the Directory Server. Since SSL is required for password change operations, this is usually give the TLS/SSL port of the Directory Server. With the <code>-zz</code> or <code>-zzz</code> for Start TLS, this can be the standard port.
-Z	Requires SSL for the connection. A secure connection is required for the password change operation. <div> NOTE <code>ldappasswd</code> also supports Start TLS encryption (<code>-zz[z]</code>).</div>
-P	Gives the full path to the certificate database which contains the CA certificate of the CA that issued the Directory Server client certificate. If the <code>ldappasswd</code> command is run on the same machine that the Directory Server is installed on, this can be <code>/etc/dirsrv/slapd-instance_name/cert8.db</code> . If this is not given, the default is the current directory.
-D	Gives the bind DN.
-w	Gives the password for the bind DN.

Parameter	Description
-a	<i>Optional.</i> Gives the old password, which is being changed.
-s	Sets the new password.

Table 7.2. ldappasswd Options

To use Start TLS, which runs the command on a non-secure port, run `ldappasswd` with the `-ZZ` option and the standard LDAP port number. The password extended change operation has the following format:

```
ldappasswd -h hostname -p standard_port -ZZ -P /path/to/cert8.db -D bindDN
-w bindPassword
-s newPassworduser [-a oldPassword]
```

Use the `-zzz` for additional certificate hostname validation.

To modify an entry's password, run `ldappasswd` like any other LDAP operation. It is not necessary to specify a *user* if the account is the same as that given in the bind DN. For example:

```
ldappasswd -h ldap.example.com -p 389 -ZZ -D
"uid=jsmith,ou=People,dc=example,dc=com"
-w rootpassword -s newpassword
```

To change the password on an entry other than the one specified in the bind credentials, run `ldappasswd` as shown below, adding the *user* DN to the operation and providing separate credentials, as follows:

```
ldappasswd -h server.example.com -p 389 -ZZ -D "cn=Directory Manager"
-w rootpassword -s newpassword "uid=jsmith,ou=People,dc=example,dc=com"
```

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.

1.4. Configuring the Account Lockout Policy

The lockout policy works in conjunction with the password policy to provide further security. The account lockout feature protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. 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.

Configuring the account lockout policy is described in the following sections:

- [Section 1.4.1, “Configuring the Account Lockout Policy Using the Console”](#)
- [Section 1.4.2, “Configuring the Account Lockout Policy Using the Command-Line”](#)

1.4.1. Configuring the Account Lockout Policy Using the Console

To set up or modify the account lockout policy for the Directory Server, do the following:

1. Select the **Configuration** tab and then the **Data** node.
2. In the right pane, select the **Account Lockout** tab.
3. To enable account lockout, select the **Accounts may be locked out** checkbox.
4. Enter the maximum number of allowed bind failures in the **Lockout account after X login failures** text box. The server locks out users who exceed the limit specified here.
5. In the **Reset failure counter after X minutes** text box, enter the number of minutes for the server to wait before resetting the bind failure counter to zero.
6. Set the interval for users to be locked out of the directory.
 - Select the **Lockout Forever** radio button to lock users out until their passwords have been reset by the administrator.
 - Set a specific lockout period by selecting the **Lockout Duration** radio button and entering the time (in minutes) in the text box.
7. Click **Save**.

1.4.2. Configuring the Account Lockout Policy Using the Command-Line

This section describes the attributes to create an account lockout policy to protect the passwords stored in the server. Use `ldapmodify` to change these attributes in the `cn=config` entry.

[Table 7.3, “Account Lockout Policy Attributes”](#) describes the attributes available to configure the account lockout policy.

Attribute Name	Definition
passwordLockout	This attribute indicates whether users are locked out of the directory after a given number of failed bind attempts. Set the number of failed bind attempts after which the user will be locked out using the

Attribute Name	Definition
	<i>passwordMaxFailure</i> attribute. Users can be locked out for a specific time or until an administrator resets the password. This attribute is set to <i>off</i> by default, meaning that users will not be locked out of the directory.
<i>passwordMaxFailure</i>	This attribute indicates the number of failed bind attempts after which a user will be locked out of the directory. This attribute takes affect only if the <i>passwordLockout</i> attribute is set to <i>on</i> . This attribute is set to 3 bind failures by default.
<i>passwordLockoutDuration</i>	This attribute indicates the time, in seconds, that users will be locked out of the directory. The <i>passwordUnlock</i> attribute specifies that a user is locked out until the password is reset by an administrator. By default, the user is locked out for 3600 seconds.
<i>passwordResetFailureCount</i>	This attribute specifies the time, in seconds, after which the password failure counter will be reset. Each time an invalid password is sent from the user's account, the password failure counter is incremented. If the <i>passwordLockout</i> attribute is set to <i>on</i> , users will be locked out of the directory when the counter reaches the number of failures specified by the <i>passwordMaxFailure</i> attribute. The account is locked out for the interval specified in the <i>passwordLockoutDuration</i> attribute, after which time the failure counter is reset to zero (0). Because the counter's purpose is to gauge when a hacker is trying to gain access to the system, the counter must continue for a period long enough to detect a hacker. However, if the counter were to increment indefinitely over days and weeks, valid users might be locked out inadvertently. The reset password failure count attribute is set 600 seconds by default.

Table 7.3. Account Lockout Policy Attributes

1.5. Managing the Password Policy in a Replicated Environment

Password and account lockout policies are enforced in a replicated environment as follows:

- Password policies are enforced on the data master.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy information in the directory is replicated:

- *passwordMinAge* and *passwordMaxAge*
- *passwordExp*
- *passwordWarning*

However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications. Account lockout counters and tiers are not replicated, either.

When configuring a password policy in a replicated environment, consider the following points:

- Warnings from the server of an impending password expiration will be issued by all replicas. This information is kept locally on each server, so if a user binds to several replicas in turn, they will be issued the same warning several times. In addition, if the user changes the password, it may take time for this information to filter to the replicas. If a user changes a password and then immediately rebinds, he may find that the bind fails until the replica registers the changes.
- 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-mastered environment.
- Entries that are created for replication (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the *passwordExpirationTime* attribute to the entry, and give it a value of 20380119031407Z (the top of the valid range).

1.6. 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 synched over to the Windows server, the Windows password policy is enforced. 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-master 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.
- 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).

See [Chapter 19, Synchronizing Red Hat Directory Server with Microsoft Active Directory](#) for more information on synchronizing Directory Server and Windows users and passwords.

2. 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 inactivate users and roles. However, when a role is inactivated, the *members of the role* are inactivated, not the role entry itself. For more information about roles in general and how roles interact with access control in particular, see [Chapter 5, Managing Entries with Roles, Class of Service, and Views](#).

- [Section 2.1, “Inactivating User and Roles Using the Console”](#)
- [Section 2.2, “Inactivating User and Roles Using the Command-Line”](#)

- [Section 2.3, “Activating User and Roles Using the Console”](#)
- [Section 2.4, “Activating User and Roles Using the Command-Line”](#)



CAUTION

The root entry (the entry corresponding to the root or sub suffix) on a database cannot be inactivated. [Chapter 2, *Creating Directory Entries*](#) has information on creating the entry for a root or sub suffix, and [Chapter 3, *Configuring Directory Databases*](#) has information on creating root and sub suffixes.

2.1. Inactivating User and Roles Using the Console

The following procedure describes inactivating a user or a role using the Console:

1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role to inactivate.

The **Edit Entry** dialog box appears.

Alternatively, select **Inactivate** from the **Object** menu.

3. Click **Account** in the left pane. The right pane states that the role or user is activate. Click the **Inactivate** to inactivate the user or role.
4. Click **OK**.

Once inactivated, the state of the object can be viewed by selecting **Inactivation State** from the **View > Display** menu. The icon of the object then appears in the right pane of the Console with a red slash through it.

2.2. Inactivating User and Roles Using the Command-Line

To inactivate a user account, use the `ns-inactivate.pl` script. The following example describes using the `ns-inactivate.pl` script to inactivate Joe Frasier's user account:

```
ns-inactivate.pl -D Directory Manager -w secretpwd -p 389 -h example.com
-I "uid=jfrasier,ou=people,dc=example,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
-D	The DN of the directory administrator.

Option Name	Description
-w	The password of the directory administrator.
-p	Port used by the server.
-h	Name of the server on which the directory resides.
-l	DN of the user account or role to inactivate.

For more information about running the `ns-inactivate.pl` script, refer to the *Directory Server Configuration, Command, and File Reference*.

2.3. Activating User and Roles Using the Console

The following procedure describes activating a user or a role using the Console:

1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role to activate.

Alternatively, select **Activate** from the **Object** menu.

The **Edit Entry** dialog box appears.

3. Click **Account** in the left pane. The right pane states that the role or user is inactivated. Click the **Activate** to activate the user or role.
4. If the user or role is a member of another inactivated role, the **Console** displays an option for viewing the inactivated roles. Click **Show Inactivated Roles** to view the list of roles to which the user or role belongs.
5. Click **OK**.

Once reactivated, the state of the object can be viewed by selecting **Inactivation State** from the **View** menu. The icon of the role or user in the right pane of the Console appears as normal. The red slash through the icon indicating it was inactive disappears.

2.4. Activating User and Roles Using the Command-Line

To activate a user account, use the `ns-activate.pl` script. The following example describes using the `ns-activate.pl` script to activate Joe Frasier's user account:

```
ns-activate.pl -D Directory Manager -w secretpwd -p 389 -h example.com
-I "uid=jfrasier,ou=people,dc=example,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
-D	The DN of the directory administrator.
-w	The password of the directory administrator.
-p	Port used by the server.
-h	Name of the server on which the directory resides.
-l	DN of the user account or role to activate.

For more information about running the `ns-activate.pl` script, refer to the *Directory Server Configuration, Command, and File Reference*.

3. Setting Resource Limits Based on the Bind DN

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.

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.

- [Section 3.1, “Setting Resource Limits Using the Console”](#)
- [Section 3.2, “Setting Resource Limits Using the Command-Line”](#)

3.1. Setting Resource Limits Using the Console

The following procedure describes setting resource limits for a user or a role using the Directory Server Console:

1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role for which to set resource limits.

The **Edit Entry** dialog box appears.

3. Click **Account** in the left pane. The right pane contains the four limits that can be set in the **Resource Limits** section.

Entering a value of -1 indicates no limit.

4. Click **OK**.

3.2. Setting Resource Limits Using the Command-Line

The following operational attributes can be set for each entry using the command-line. Use `ldapmodify` to add the following attributes to the entry:

Attribute	Description
<code>nsLookThroughLimit</code>	Specifies how many entries are examined for a search operation. Giving this attribute a value of -1 indicates that there is no limit.
<code>nsSizeLimit</code>	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.
<code>nsTimeLimit</code>	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.
<code>nsIdleTimeout</code>	Specifies the time a connection to the server can be idle before the connection is dropped. The value is given in seconds. Giving this attribute a value of -1 indicates that there is no limit.

For example, this sets the size limit for Barbara Jensen by using `ldapmodify`¹ to modify her entry:

```
ldapmodify -h myserver -p 389 -D "cn=directory manager" -w secretpwd

dn: uid=bjensen,ou=people,dc=example,dc=com
changetype: modify
add:nsSizeLimit
nsSizeLimit: 500
```

The `ldapmodify` statement adds the `nsSizeLimit` attribute to Babs Jensen's entry and gives it a search return size limit of 500 entries.

Managing Replication

Replication is the mechanism by which directory data is automatically copied from one Red Hat Directory Server instance to another; it is an important mechanism for extending the directory service beyond a single server configuration. This chapter describes the tasks to be performed on the master and consumer servers to set up single-master replication, multi-master replication, and cascading replication.

1. Replication Overview

Replication is the mechanism by which directory data is automatically copied from one Directory Server to another. Updates of any kind — entry additions, modifications, or even deletions — are automatically mirrored to other Directory Servers using replication. This section contains information on the following replication concepts:

- [Section 1.1, “What Directory Units Are Replicated”](#)
- [Section 1.2, “Read-Write and Read-Only Replicas”](#)
- [Section 1.3, “Suppliers and Consumers”](#)
- [Section 1.4, “Changelog”](#)
- [Section 1.5, “Replication Identity”](#)
- [Section 1.6, “Replication Agreement”](#)
- [Section 1.7, “Compatibility with Earlier Versions of Directory Server”](#)

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, “Creating and Maintaining Databases”](#).

1.2. Read-Write and Read-Only Replicas

A database that participates in replication is called a *replica*. There are two kinds of replicas: read-write or read-only. A *read-write replica* contains master copies of directory information and can be updated. A *read-only replica* services read, search, and compare requests, but refers all update operations to read-write replicas. A server can hold any number of read-only or

read-write replicas.

1.3. Suppliers and Consumers

A server that holds a replica that is copied to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is copied from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica, with two exceptions:

- In the case of cascading replication, the hub server holds a read-only replica that it supplies to consumers. [Section 2.3, “Cascading Replication”](#) has more information.
- In the case of multi-master replication, the *masters* are both suppliers and consumers for the same information. For more information, see [Section 2.2, “Multi-Master Replication”](#).

Replication is always initiated by the supplier server, never by the consumer (*supplier-initiated replication*). Supplier-initiated replication allows a supplier server to be configured to push data to multiple consumer servers.

1.4. Changelog

Every supplier server maintains a *changelog*, a record of all changes that a supplier or hub needs to send to its consumers. A changelog is a special kind of database that describes the modifications that have occurred on a replica. The supplier server then replays these modifications to the replicas stored on consumer servers or to other suppliers, in the case of multi-master replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the changelog.

In Directory Server, the changelog is only intended for internal use by the server. For other applications to read the changelog, use the Retro Changelog Plug-in, as described in [Section 16, “Using the Retro Changelog Plug-in”](#).

1.5. Replication Identity

When replication occurs between two servers, the replication process uses a special entry, called the *replication manager* entry, to identify replication protocol exchanges and to control access to the directory data. The replication manager entry, or any entry used during replication, must meet the following criteria:

- It is created on the consumer server (or hub) and *not* on the supplier server.
- Create this entry on *every* server that receives updates from another server, meaning on every hub or dedicated consumer.

- When a replica is configured as a consumer or hub (a replica which receives updates from another server), this entry must be specified as the one authorized to perform replication updates.
- The replication agreement is created on the supplier server, the DN of this entry must be specified in the replication agreement.
- The supplier bind DN entry must not be part of the replicated database for security reasons.
- This entry, with its special user profile, bypasses all access control rules defined on the consumer server for the database involved in that replication agreement.



NOTE

In the Directory Server Console, this replication manager entry is referred to as the *supplier bind DN*, which may be misleading because the entry does not actually exist on the supplier server. It is called the supplier bind DN because it is the entry which the supplier uses to bind to the consumer. This entry actually exists, then, on the consumer.

For more information on creating the replication manager entry, see [Section 3, “Creating the Supplier Bind DN Entry”](#).

1.6. Replication Agreement

Directory Servers use replication agreements to define their replication configuration. A replication agreement describes replication between *one* supplier and *one* consumer only. The agreement is configured on the supplier server and must specify all required replication information:

- The database to be replicated.
- The consumer server to which the data is pushed.
- The days and times during which replication can occur.
- The DN and credentials that the supplier server must use to bind (the replication manager entry or supplier bind DN).
- How the connection is secured (SSL, client authentication).
- Any attributes that will not be replicated (fractional replication).

1.7. Compatibility with Earlier Versions of Directory Server

The replication mechanism in Directory Server 8.0 is different from the mechanism used in 4.x

of Directory Server. Compatibility is provided through two Directory Server plug-ins:

- *Legacy Replication Plug-in.* The Legacy Replication Plug-in makes a Directory Server 8.0 instance behave as a 4.x Directory Server in a consumer role. For information on how to implement legacy replication using this plug-in, see [Section 15, “Replication with Earlier Releases”](#).
- *Retro Changelog Plug-in.* The Retro Changelog Plug-in can be used for a Directory Server supplier to maintain a 4.x-style changelog. This is sometimes necessary for legacy applications that have a dependency on the Directory Server 4.x changelog format because they read information from the changelog. For more information on the Retro Changelog Plug-in, see [Section 16, “Using the Retro Changelog Plug-in”](#).

2. Replication Scenarios

This section describes the most commonly used replication scenarios:

- [Section 2.1, “Single-Master Replication”](#)
- [Section 2.2, “Multi-Master Replication”](#)
- [Section 2.3, “Cascading Replication”](#)

These basic strategies can be combined in a variety of ways to create the best replication environment.



NOTE

Whatever replication scenario is implemented, consider schema replication. To avoid conflict resolution loops, the Referential Integrity Plug-in should only be enabled on one supplier replica in a multi-master replication environment. The plug-in is off by default.

2.1. Single-Master Replication

In the simplest replication scenario, the master copy of directory data is held in a single read-write replica on one server called the *supplier server*. The supplier server also maintains changelog for this replica. On another server, called the *consumer server*, there can be multiple read-only replicas. Such scenarios are called *single-master configurations*. [Figure 8.1, “Single-Master Replication”](#) shows an example of single-master replication.

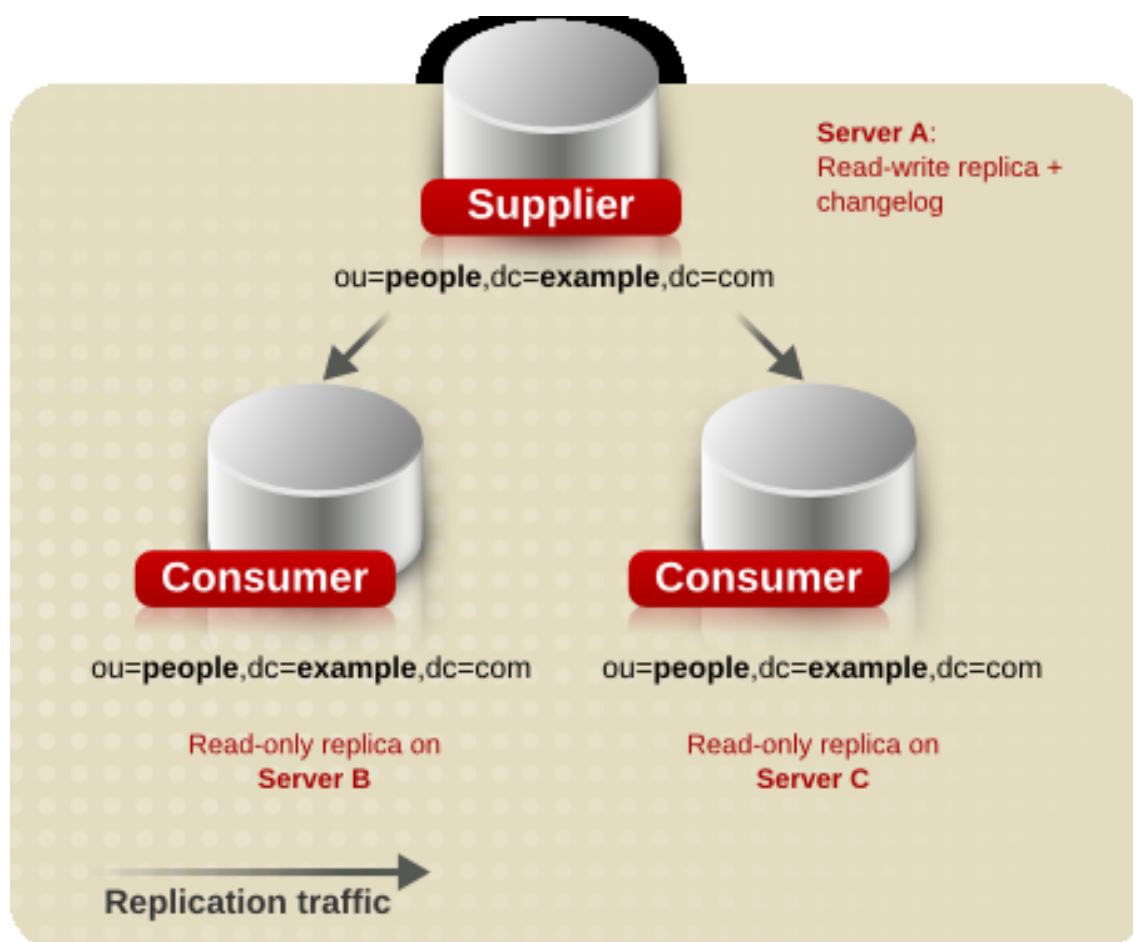


Figure 8.1. Single-Master Replication

In this particular configuration, the `ou=people,dc=example,dc=com` suffix receives a large number of search requests. Therefore, to distribute the load, this tree, which is mastered on server A, is replicated to two read-only replicas located on server B and server C.

For information on setting up a single-master replication environment, see [Section 4](#), “*Configuring Single-Master Replication*”.

2.2. Multi-Master Replication

Directory Server also supports complex replication scenarios in which the same suffix (database) can be mastered on many servers. This suffix is held in a read-write replica on each server. This means that each server maintains a changelog for the read-write replica.

This type of configuration can work with any number of consumer servers. Each consumer server holds a read-only replica. The consumers can receive updates from all the suppliers. The consumers also have referrals defined for all the suppliers to forward any update requests that the consumers receive. Such scenarios are called *multi-master configurations*.

Figure 8.2, “Multi-Master Replication (Two Masters)” shows an example of multi-master replication scenario with two supplier servers and two consumer servers.

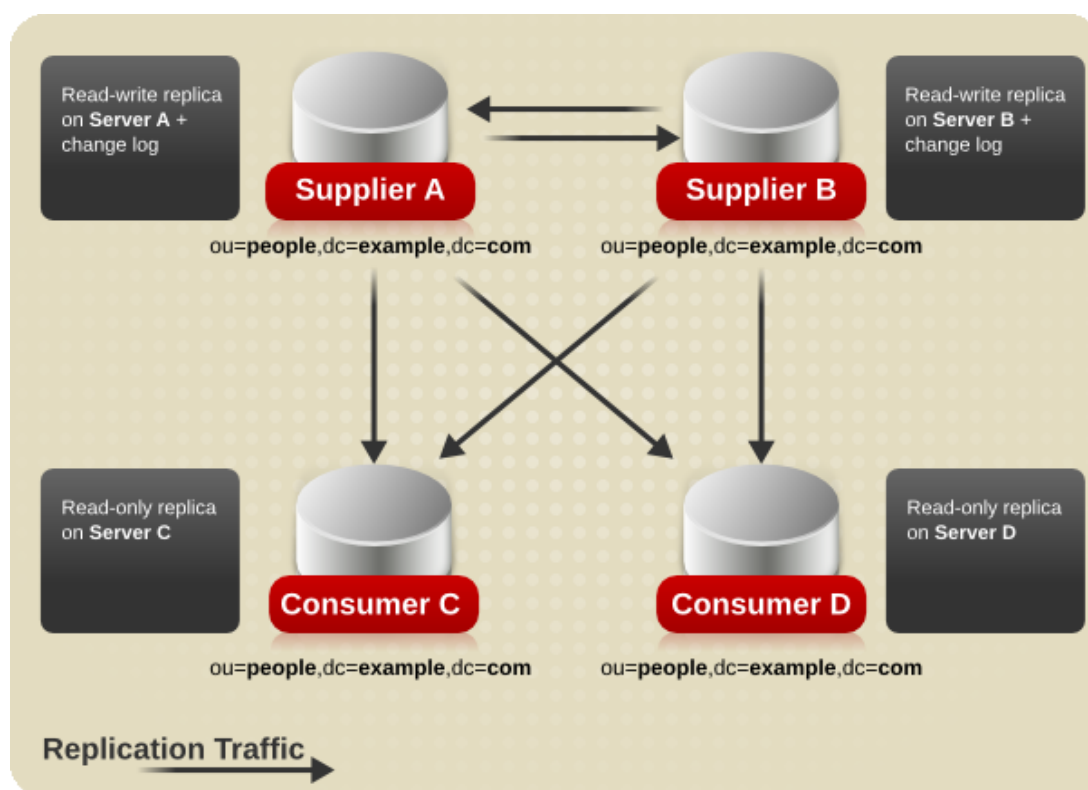


Figure 8.2. Multi-Master Replication (Two Masters)

Figure 8.3, “Multi-Master Replication (Four Masters)” shows a sample of multi-master replication scenario with four supplier servers and eight consumer servers. In this sample setup, each supplier server is configured with ten replication agreements to feed data to two other supplier servers and all eight consumer servers.

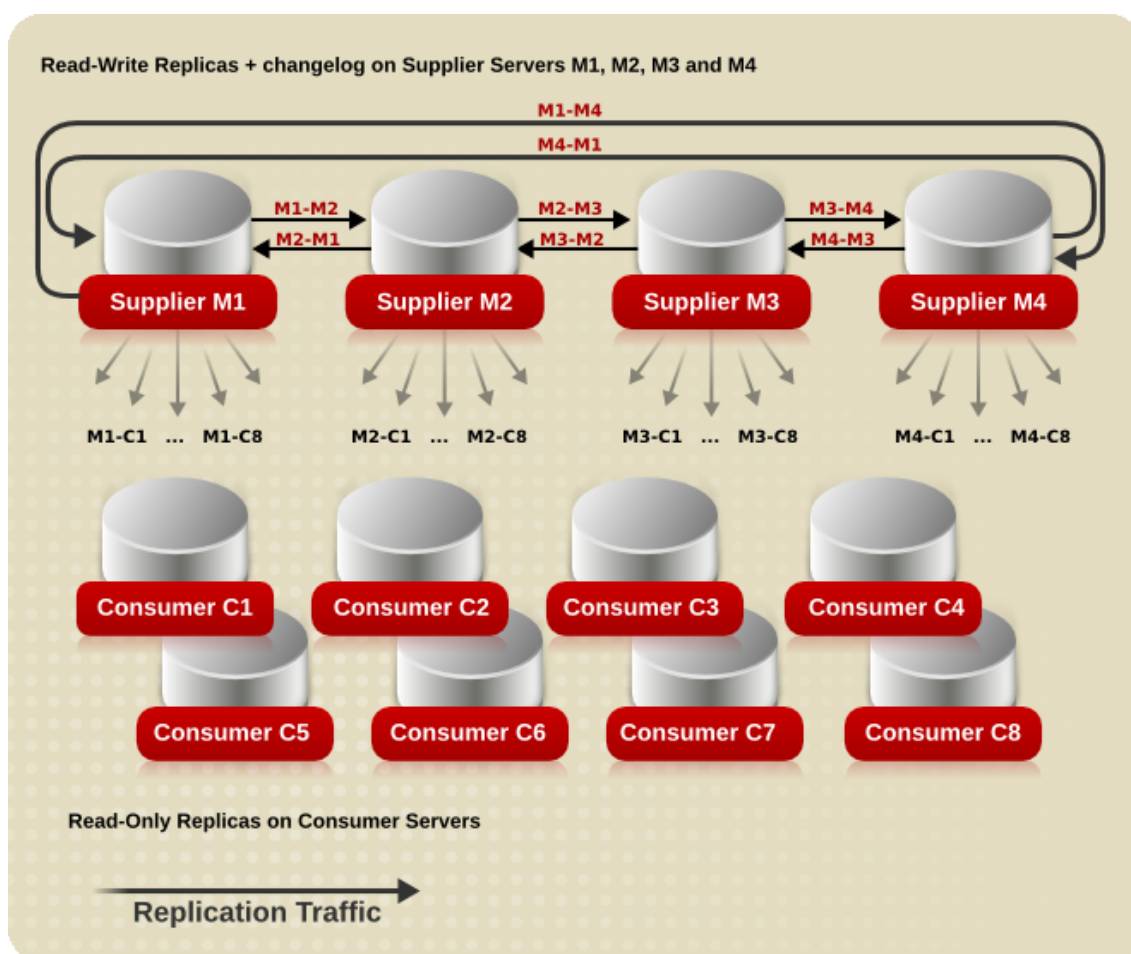


Figure 8.3. Multi-Master Replication (Four Masters)

Multi-master configurations have the following advantages:

- Automatic write failover when one supplier is inaccessible.
- Updates are made on a local supplier in a geographically distributed environment.



NOTE

The speed that replication proceeds depends on the speed of the network. Plan changes and directory configuration accordingly, and realize that changes to one directory may not be quickly replicated to other directories over slow links, such as wide-area networks, in geographically-distributed environments.

For the procedure to set up multi-master replication, see [Section 5, “Configuring Multi-Master](#)

Replication”.

2.3. Cascading Replication

In a cascading replication scenario, one server, a *hub*, acts both as a consumer and a supplier. It holds a read-only replica and maintains a changelog, so it receives updates from the supplier server that holds the master copy of the data and, in turn, supplies those updates to the consumer. Cascading replication is very useful for balancing heavy traffic loads or to keep master servers based locally in geographically-distributed environments.

Figure 8.4, “Cascading Replication” shows an example of a simple cascading replication scenario, though it is possible to create more complex scenarios with several hub servers.

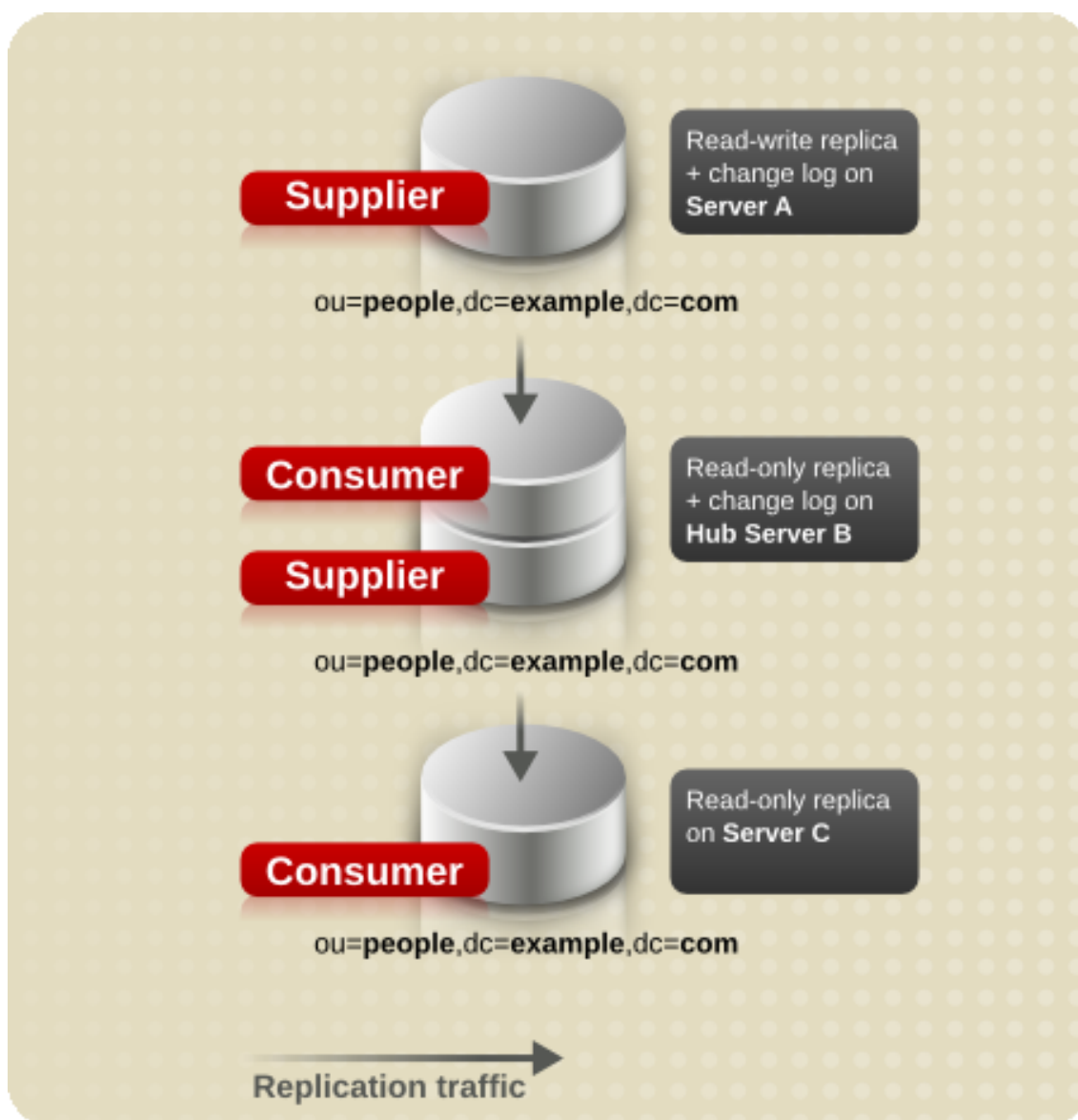


Figure 8.4. Cascading Replication

For information on setting up cascading replication, see [Section 6, “Configuring Cascading Replication”](#).



NOTE

Multi-master and cascading replication can be combined. For example, in the multi-master scenario illustrated in [Figure 8.2, “Multi-Master Replication \(Two Masters\)”](#), server C and server D could be hub servers that would replicate to any number of consumer servers.

3. Creating the Supplier Bind DN Entry

A critical part of setting up replication is to create the entry, called the replication manager or supplier bind DN entry, that the suppliers use to bind to the consumer servers to perform replication updates.

The supplier bind DN must meet the following criteria:

- It must be unique.
- It must be created on the consumer server (or hub) and *not* on the supplier server.
- It must correspond to an actual entry on the consumer server.
- It must be created on *every* server that receives updates from another server.
- It must not be part of the replicated database for security reasons.
- It must be defined in the replication agreement on the supplier server.

For example, the entry `cn=Replication Manager,cn=config` can be created under the `cn=config` tree on the consumer server. This would be the supplier bind DN that all supplier servers would use to bind to the consumer to perform replication operations.



NOTE

Avoid creating simple entries under the `cn=config` entry in the `dse.ldif` file. The `cn=cn=config` entry in the simple, flat `dse.ldif` configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, and particularly entries that are likely to be updated frequently, are stored under `cn=config`, performance will suffer. However, although Red Hat recommends not storing simple user entries under `cn=config` for performance reasons, it can be useful to store special user entries such as the Directory

Manager entry or replication manager (supplier bind DN) entry under `cn=config` since this centralizes configuration information.

On each server that acts as a consumer in replication agreements, create a special entry that the supplier will use to bind to the consumers. Make sure to create the entry with the attributes required by the authentication method specified in the replication agreement.

1. Stop the Directory Server. If the server is not stopped, the changes to the `dse.ldif` file will not be saved. See [Section 3, “Starting and Stopping Servers”](#) for more information on stopping the server.
2. Create a new entry, such as `cn=replication manager,cn=config`, in the `dse.ldif` file.
3. Specify a `userPassword` attribute-value pair.
4. If password expiration policy is enabled or ever will be enabled, disable it on the replication manager entry to prevent replication from failing due to passwords expiring. To disable the password expiration policy on the `userPassword` attribute, add the `passwordExpirationTime` attribute with a value of `20380119031407Z`, which means that the password will never expire.
5. Restart the Directory Server. See [Section 3, “Starting and Stopping Servers”](#) for more information on starting the server.

The final entry should resemble this example:

```
dn: cn=replication manager,cn=config
objectClass: inetorgperson
objectClass: person
objectClass: top
cn: replication manager
sn: RM
userPassword: password
passwordExpirationTime: 20380119031407Z
```

When configuring a replica as a consumer, use the DN of this entry to define the supplier bind DN.

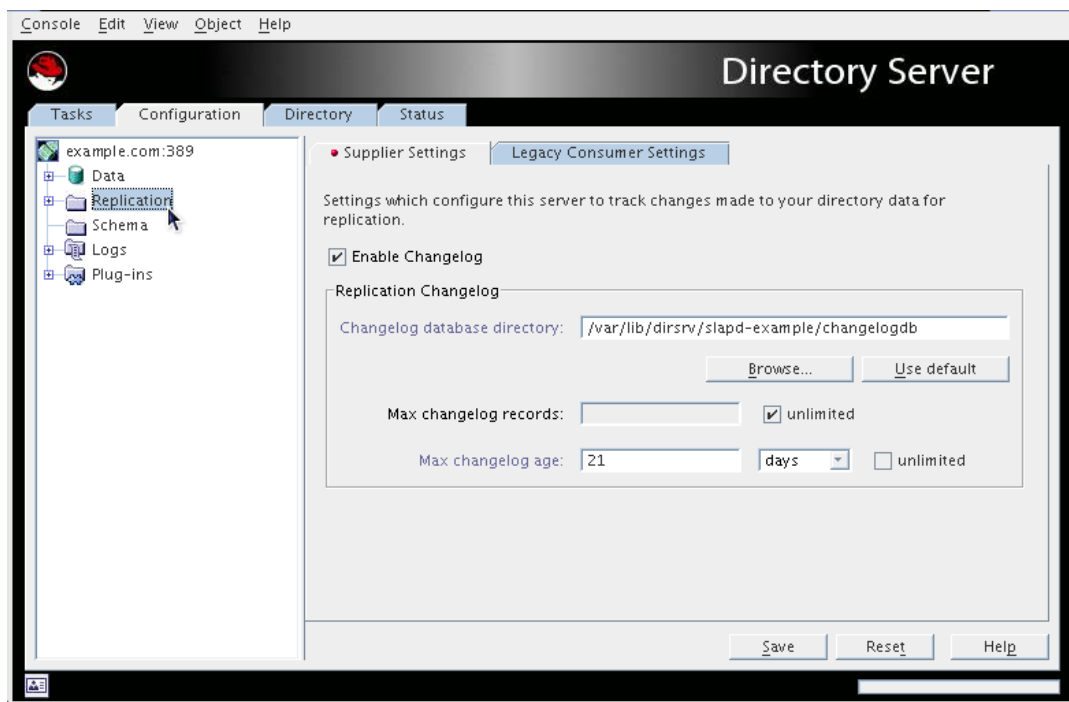
4. Configuring Single-Master Replication

To set up single-master replication such as the configuration shown in [Figure 8.1, “Single-Master Replication”](#), between supplier server A, which holds a read-write replica, and the two consumers server B and server C, which each hold a read-only replica, there are two major steps:

- [Section 4.1, “Configuring the Read-Write Replica on the Supplier Server”](#)
- [Section 4.2, “Configuring the Read-Only Replica on the Consumer”](#)
- [Section 4.3, “Create the Replication Agreement”](#)

4.1. Configuring the Read-Write Replica on the Supplier Server

1. Specify the supplier settings for the server.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, select the **Replication** folder.
 - c. In the right-hand side of the window, select the **Supplier Settings** tab.



- d. Check the **Enable Changelog** checkbox.

This activates all of the fields in the pane below that were previously grayed out.
- e. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.
- f. Set the changelog parameters for the number and age of the log files.

Clear the unlimited checkboxes to specify different values.
- g. Click **Save**.

2. Specify the replication settings required for a read-write replica.

- a. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

The **Replica Settings** tab opens in the right-hand side of the window.

- b. Check the **Enable Replica** checkbox.
- c. In the **Replica Role** section, select the **Single Master** radio button.
- d. In the **Common Settings** section, specify a **Replica ID**, which is an integer between 1 and 65534, inclusive.

The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

- e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

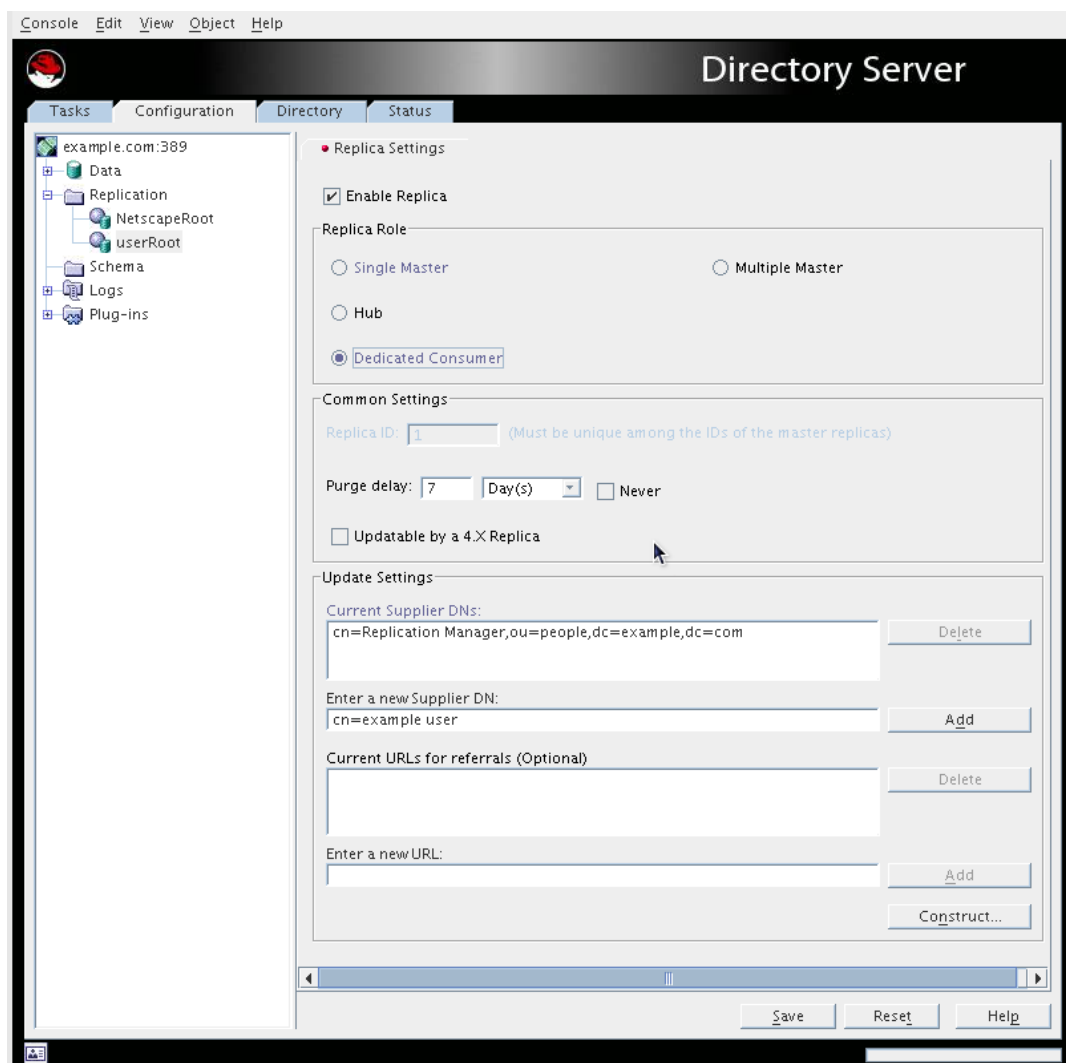
The purge delay is how often the state information stored in the replicated entries is deleted.

- f. Click **Save**.

4.2. Configuring the Read-Only Replica on the Consumer

1. Create the database for the read-only replica if it does not exist. See [Section 1.1, “Creating Suffixes”](#) for instructions on creating suffixes.
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in [Section 3, “Creating the Supplier Bind DN Entry”](#).
3. Specify the replication settings required for a read-only replica.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

The **Replica Settings** tab for that database opens in the right-hand side of the window.



- c. Check the **Enable Replica** checkbox.
- d. In the **Replica Role** section, select the **Dedicated Consumer** radio button.
- e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

This option indicates how often the state information stored in the replicated entries is purged.

- f. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control.



NOTE

There can be multiple supplier bind DN's per consumer but only one supplier DN per replication agreement.

- g. Specify the URL for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients bind over a regular connection; this has a URL in the form `ldap://hostname:port`. For clients to bind to the supplier using SSL, use this field to specify a referral of the form `ldaps://hostname:port`, where the `s` in `ldaps` indicates a secure connection.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration.

4.3. Create the Replication Agreement

Create one replication agreement for each read-only replica. For example, in the scenario illustrated in [Figure 8.1, “Single-Master Replication”](#), server A has two replication agreements, one for server B and one for server C.

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.

Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.
3. In the **Source and Destination** screen, fill in the URL for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.

Source and Destination

Provide server and content information:

Supplier
example.com:389

Consumer
example-ldap.com:1389 Other...

Connection
☐ Using encrypted SSL connection
Authenticate using:
☐ SSL client authentication
☒ Simple authentication
Bind as: cn=Replication Manager,ou=people,dc=example
Password: *****

Subtree:
dc=example, dc=com

Back Next Cancel Help

- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.
- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind DN and password.

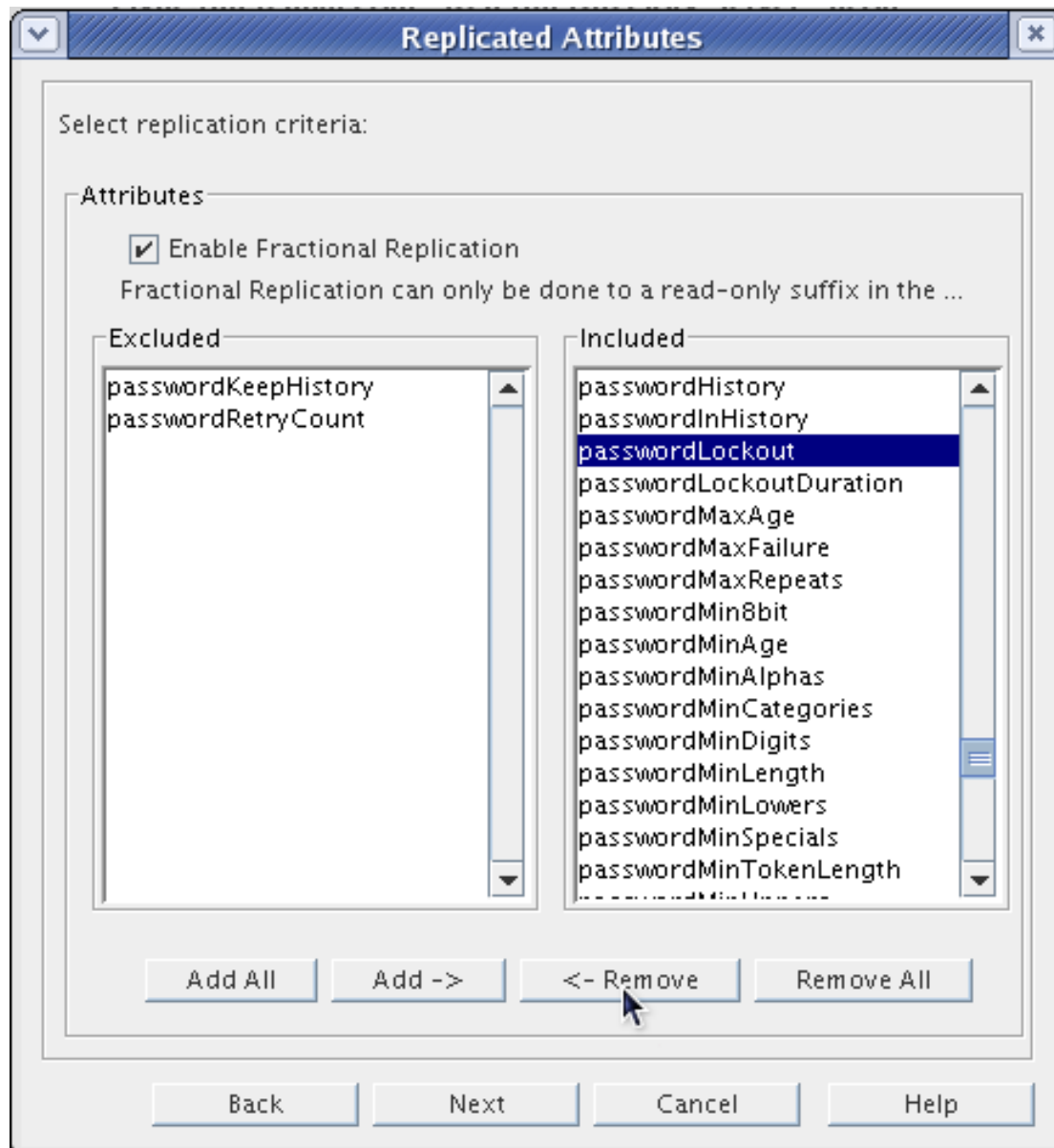


NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

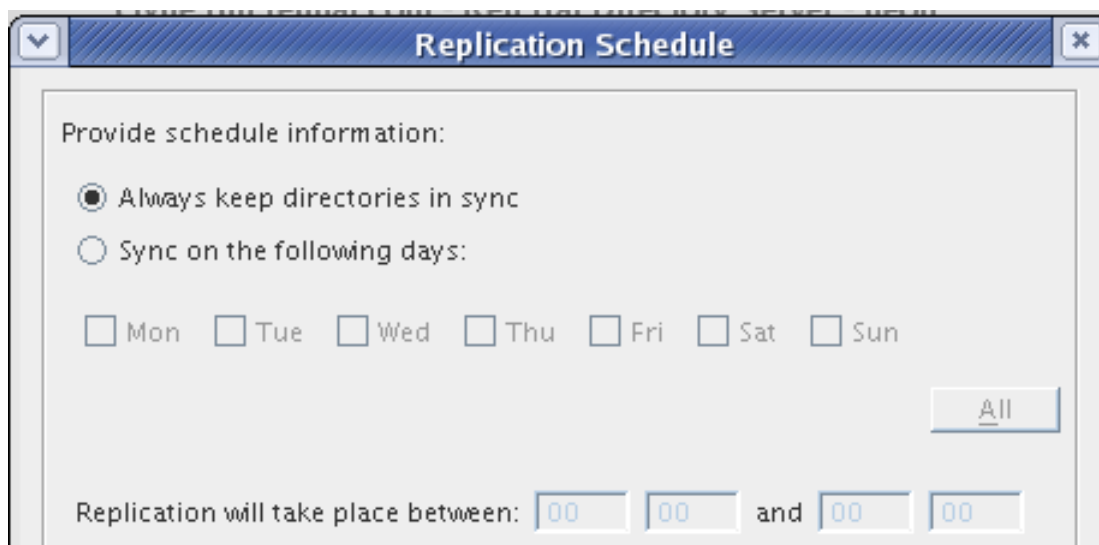
Hit **Next**.

4. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** checkbox. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.

**NOTE**

To safeguard against potential integrity problems, the consumer in fractional replication must be a dedicated consumer, not a multi-master supplier or hub. This is not enforced at the time the replication agreement is made, but replication will fail if the consumer is not a read-only replica.

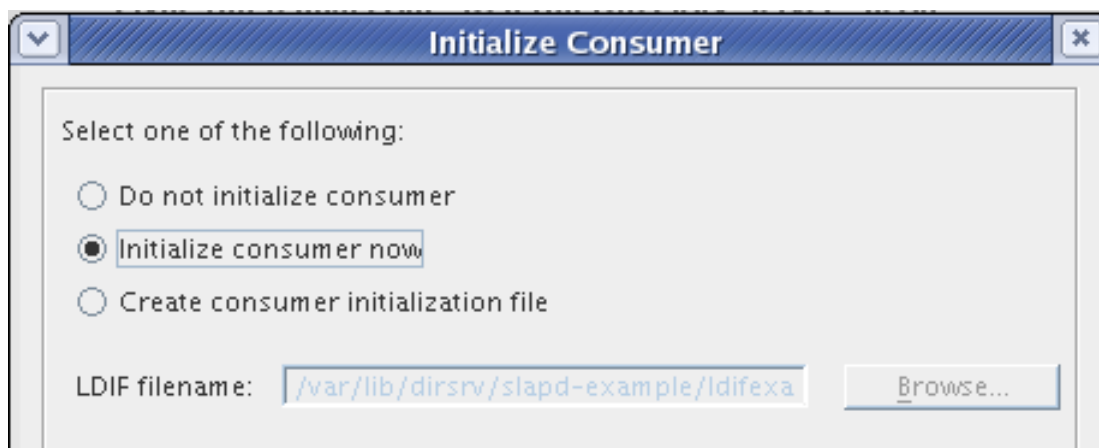
5. Set the schedule for when replication runs. By default, replication runs continually.



The **Replication Schedule** dialog box has a title bar with a dropdown arrow and a close button. The main area contains the text "Provide schedule information:" followed by two radio buttons: "Always keep directories in sync" (selected) and "Sync on the following days:". Below the second radio button are seven checkboxes for the days of the week: Mon, Tue, Wed, Thu, Fri, Sat, and Sun. To the right of these checkboxes is a button labeled "All". At the bottom, the text "Replication will take place between:" is followed by two time input fields (each showing "00"), the word "and", and another two time input fields (each showing "00").

Hit **Next**.

- Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see [Section 10, "Initializing Consumers"](#).



The **Initialize Consumer** dialog box has a title bar with a dropdown arrow and a close button. The main area contains the text "Select one of the following:" followed by three radio buttons: "Do not initialize consumer", "Initialize consumer now" (selected), and "Create consumer initialization file". Below these is a text field labeled "LDIF filename:" containing the path `/var/lib/dirsrv/slapd-example/ldifexa`. To the right of the text field is a button labeled "Browse...".



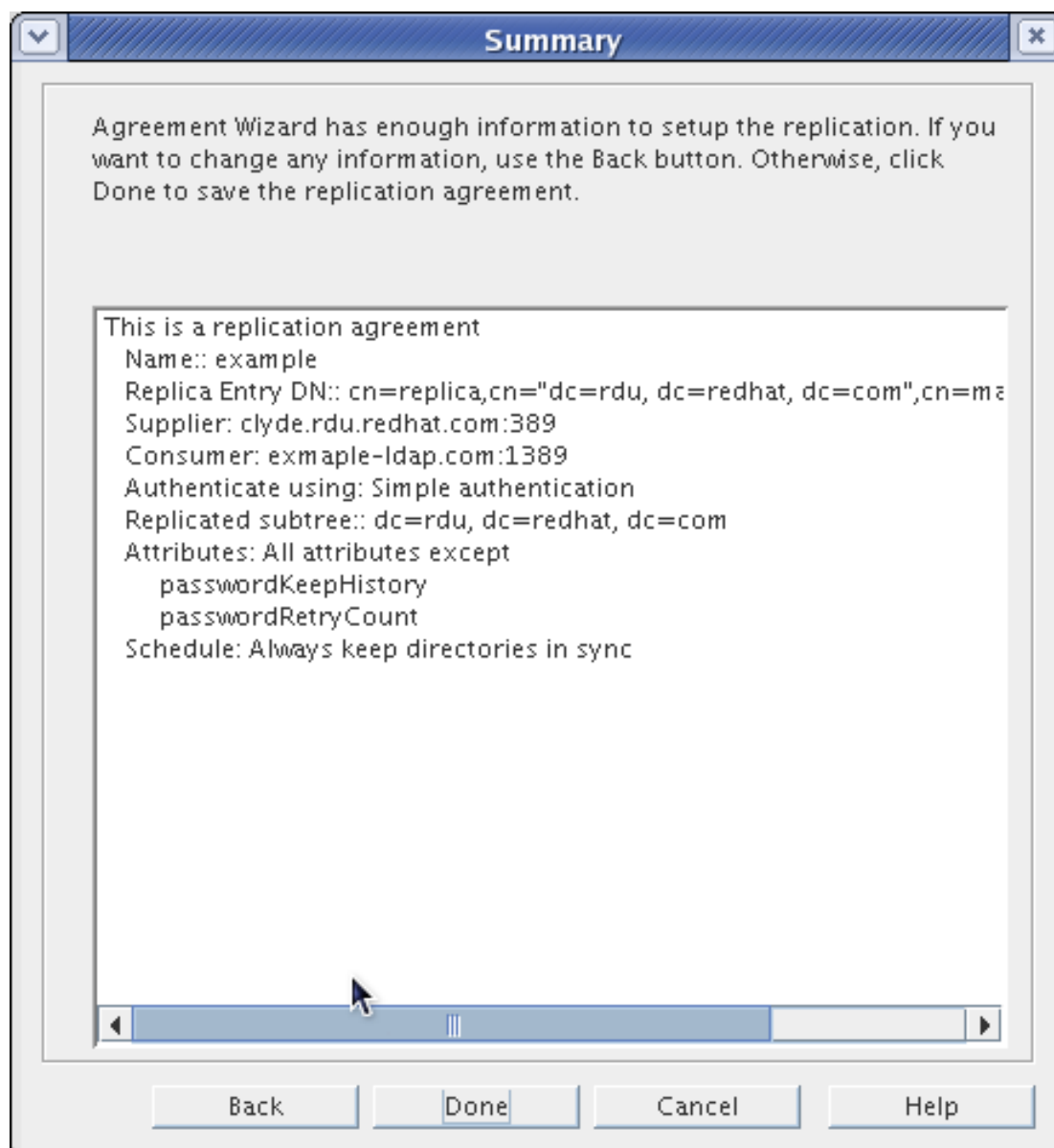
NOTE

Replication *will not* begin until the consumer is initialized.

Hit **Next**.

- The final screen shows the settings for the replication agreement, as it will be included in the

dse.ldif file. Hit **Done** to save the agreement.



The replication agreement is set up.



NOTE

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

5. Configuring Multi-Master Replication

This section provides information on configuring multi-master replication. In a multi-master configuration, many suppliers can accept updates, synchronize with each other, and update all consumers. The consumers can send referrals for updates to all masters.

Directory Server supports 4-way multi-master replication. To set up multi-master replication such as the configuration shown in [Figure 8.3, “Multi-Master Replication \(Four Masters\)”](#), set up all of the consumers first, then set up the suppliers, and last, initialize all of the databases:

- [Section 5.1, “Configuring the Read-Write Replicas on the Supplier Servers”](#)
- [Section 5.2, “Configuring the Read-Only Replicas on the Consumer Servers”](#)
- [Section 5.3, “Setting up the Replication Agreements”](#)
- [Section 5.4, “Preventing Monopolization of the Consumer in Multi-Master Replication”](#)



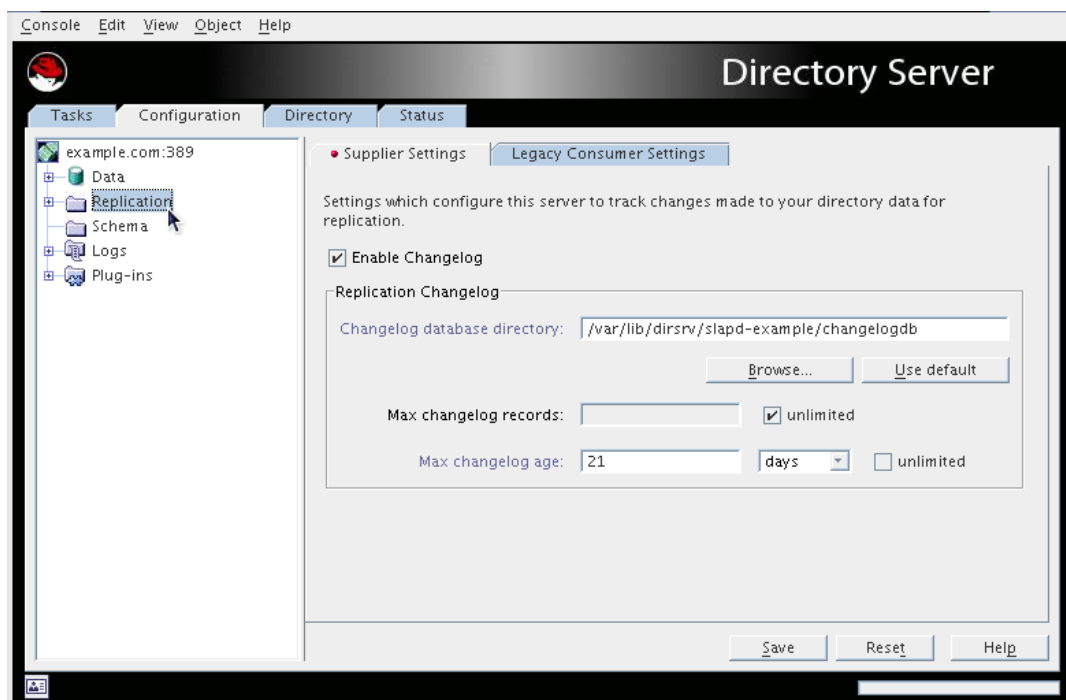
TIP

More than 10 databases running with replication or more than 20 replication agreements on a supplier can cause performance degradation. To support that many consumers, introduce hub replicas between the suppliers and consumers. See [Section 6, “Configuring Cascading Replication”](#).

5.1. Configuring the Read-Write Replicas on the Supplier Servers

Set up each supplier server. The first supplier configured should be used to initialize the other suppliers in the multi-master replication environment.

1. Specify the supplier settings for the server.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, select the **Replication** folder.
 - c. In the right-hand side of the window, select the **Supplier Settings** tab.



- d. Check the **Enable Changelog** checkbox.

This activates all of the fields in the pane below that were previously grayed out.

- e. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.
- f. Set the changelog parameters for the number and age of the log files.

Clear the unlimited checkboxes to specify different values.

- g. Click **Save**.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. This is the special entry that the other suppliers will use to bind to this supplier, as in other supplier-consumer relationships. This is described in [Section 3, "Creating the Supplier Bind DN Entry"](#).

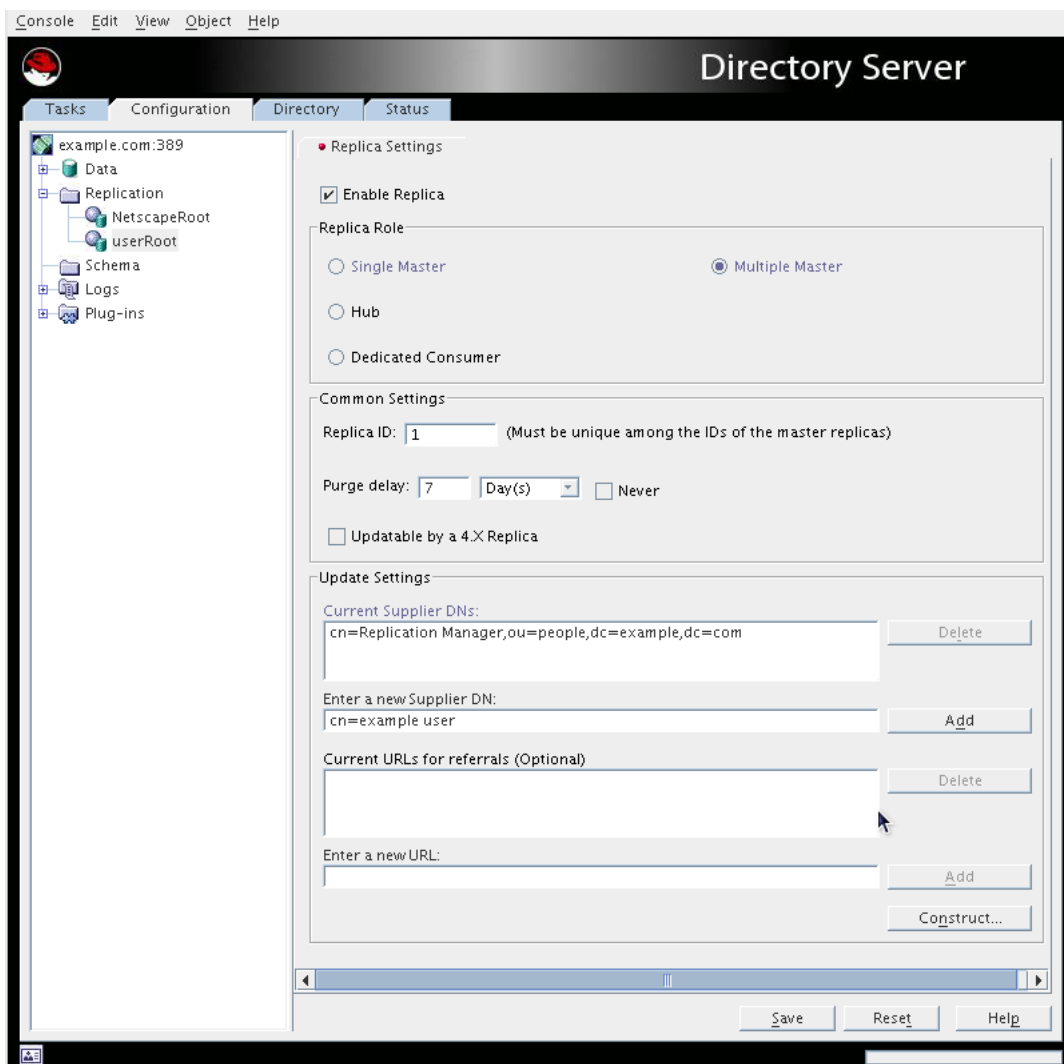


NOTE

For multi-master replication, it is necessary to create this supplier bind DN on the supplier servers as well as the consumers because the suppliers act as both consumer and supplier to the other supplier servers.

3. Specify the replication settings for the multi-mastered read-write replica.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

The **Replica Settings** tab for that database opens in the right-hand side of the window.



- c. Check the **Enable Replica** checkbox.
- d. In the **Replica Role** section, select the **Multi-Master** radio button.
- e. In the **Common Settings** section, specify a **Replica ID**, which is an integer between 1 and 65534, inclusive.

The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.
- f. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

The purge delay is how often the state information stored in the replicated entries is deleted.

- g. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.



NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

- h. Specify the URL for any supplier servers to which to refer updates, such as the other suppliers in the multi-master replication set. Only specify the URL for the supplier server.

For clients to bind using SSL, specify a URL beginning with `ldaps://`.

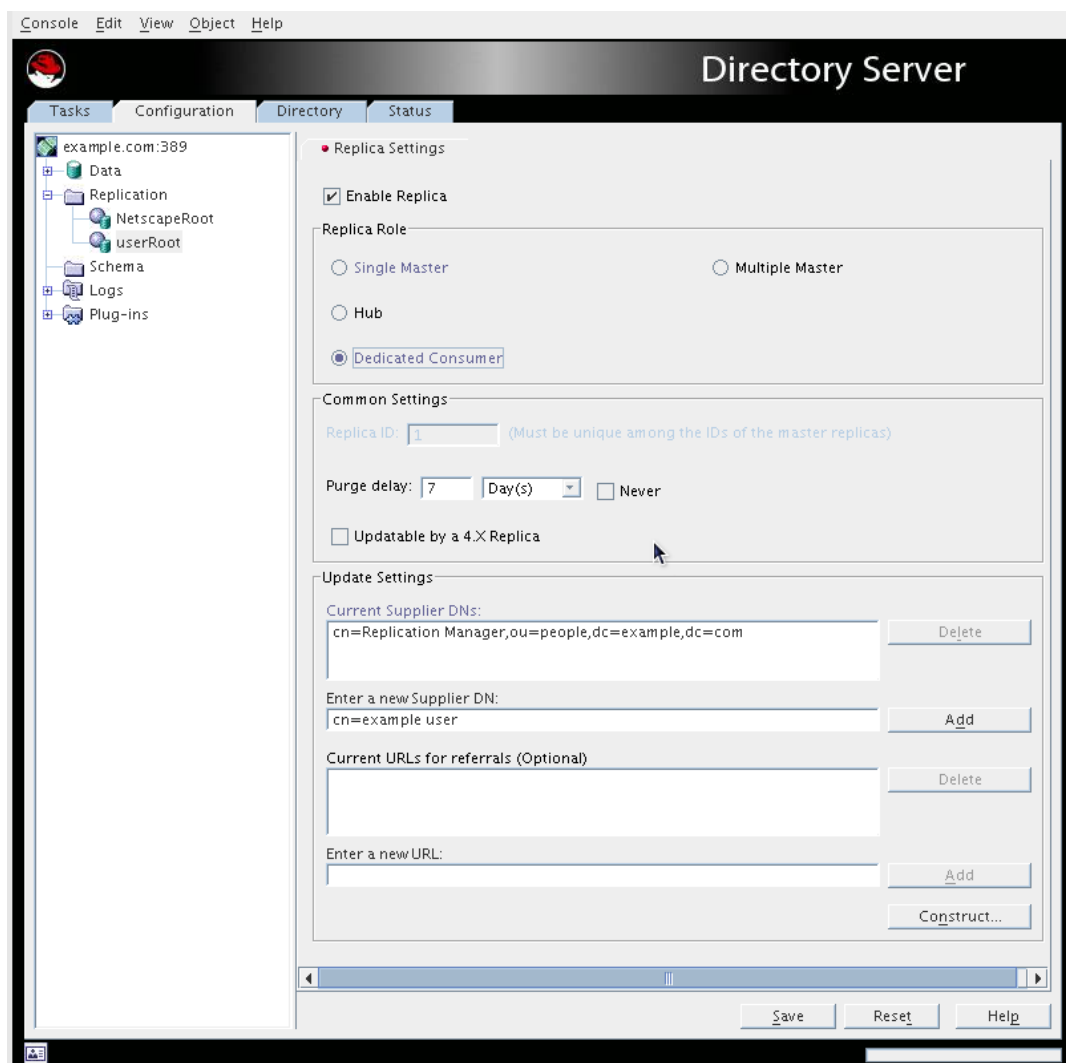
- i. Click **Save**.

5.2. Configuring the Read-Only Replicas on the Consumer Servers

First, configure every consumer.

1. Create the database for the read-only replica if it does not exist. See [Section 1.1, “Creating Suffixes”](#) for instructions on creating suffixes.
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in [Section 3, “Creating the Supplier Bind DN Entry”](#).
3. Specify the replication settings required for a read-only replica.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

The **Replica Settings** tab for that database opens in the right-hand side of the window.



- c. Check the **Enable Replica** checkbox.
- d. In the **Replica Role** section, select the **Dedicated Consumer** radio button.
- e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

This option indicates how often the state information stored in the replicated entries is purged.

- f. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

**NOTE**

There can be multiple supplier bind DN's per consumer but only one supplier DN per replication agreement.

- g. Specify the URL for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients bind over a regular connection; this has a URL in the form `ldap://hostname:port`. For clients to bind to the supplier using SSL, use this field to specify a referral of the form `ldaps://hostname:port`, where the `s` in `ldaps` indicates a secure connection.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration.

5.3. Setting up the Replication Agreements

First set up replication agreements on a single supplier, the data master, between the other multi-master suppliers, and initialize all of the other suppliers.

Then create replication agreements for all other suppliers in the multi-master replication set, but *do not* reinitialize any of the suppliers.

Then create replication agreements for all of the consumers from the single data master, and initialize the consumers.

Then create replication agreements for all of the consumers from for all of the other suppliers, but *do not* reinitialize any of the consumers.

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.

Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.
3. In the **Source and Destination** screen, fill in the URL for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.

Source and Destination

Provide server and content information:

Supplier
example.com:389

Consumer
example-ldap.com:1389 Other...

Connection
☐ Using encrypted SSL connection
Authenticate using:
☐ SSL client authentication
☒ Simple authentication
Bind as: cn=Replication Manager,ou=people,dc=example
Password: *****

Subtree:
dc=example, dc=com

Back Next Cancel Help

- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.
- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind DN and password.

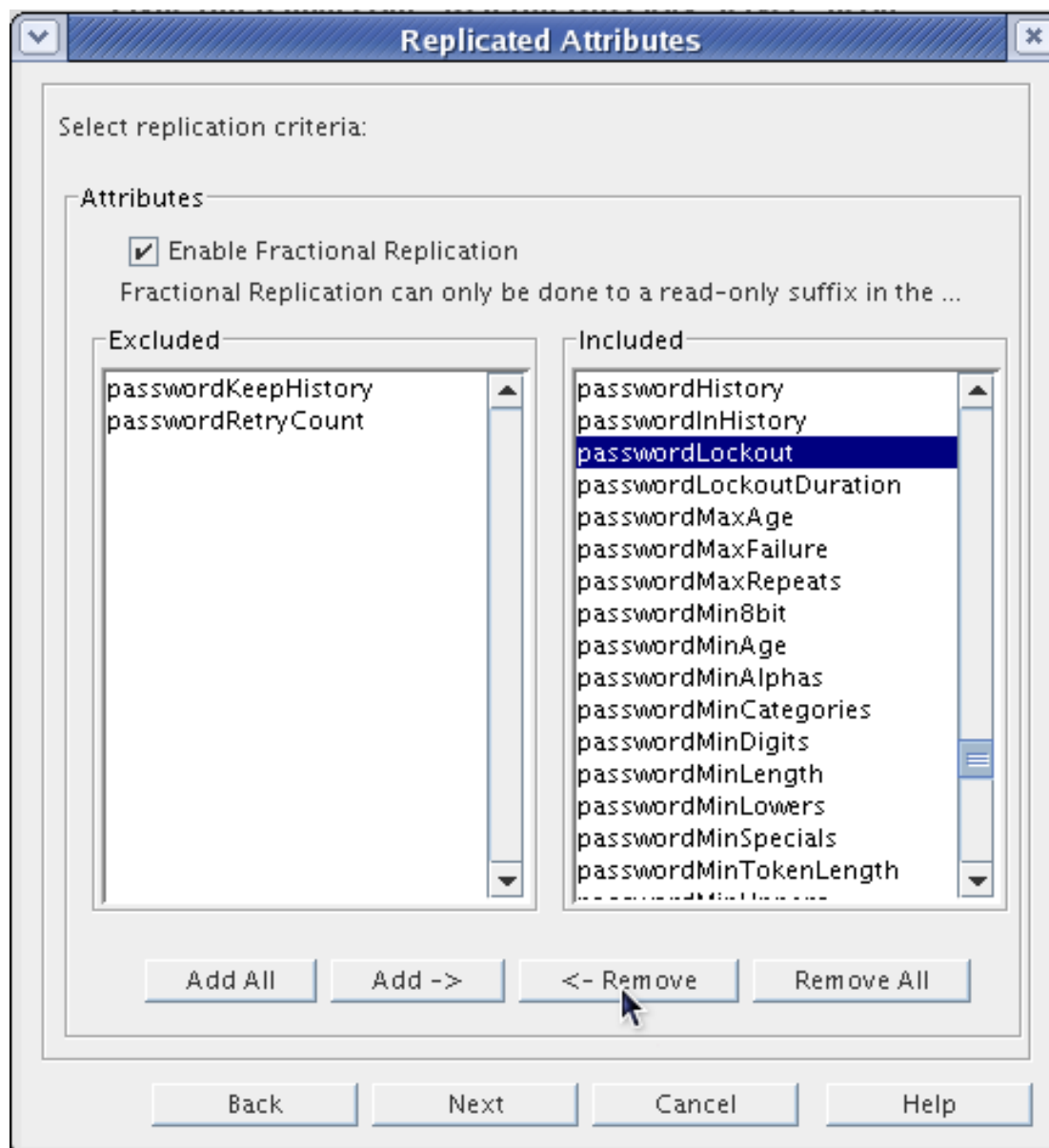


NOTE

If attribute encryption is enabled, a secure connection is required for the encrypted attributes to be replicated.

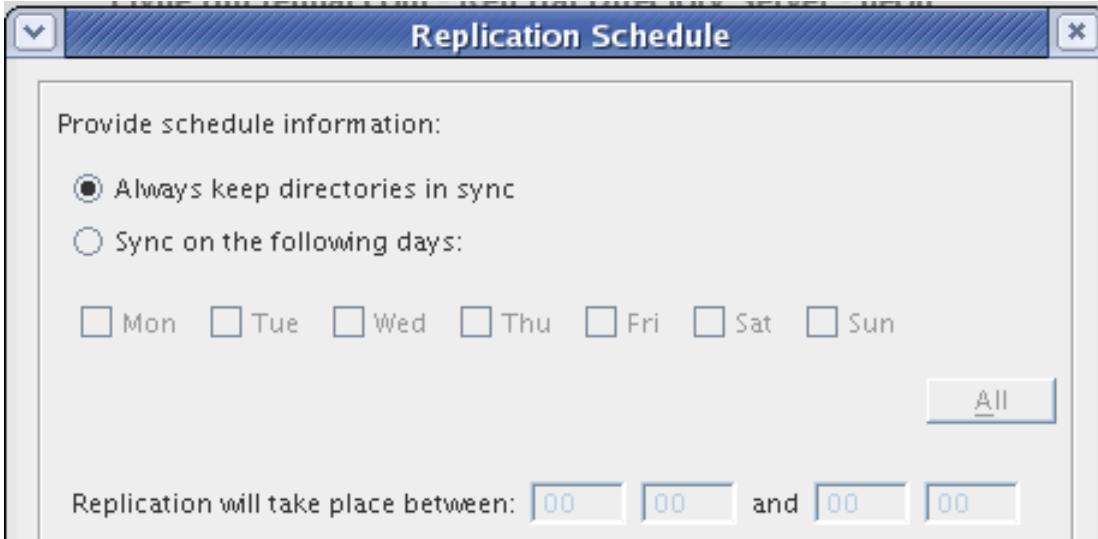
Hit **Next**.

4. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** checkbox. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.

**NOTE**

To safeguard against potential integrity problems, the consumer in fractional replication must be a dedicated consumer, not a multi-master supplier or hub. This is not enforced at the time the replication agreement is made, but replication will fail if the consumer is not a read-only replica.

5. Set the schedule for when replication runs. By default, replication runs continually.



Replication Schedule

Provide schedule information:

☒ Always keep directories in sync

☐ Sync on the following days:

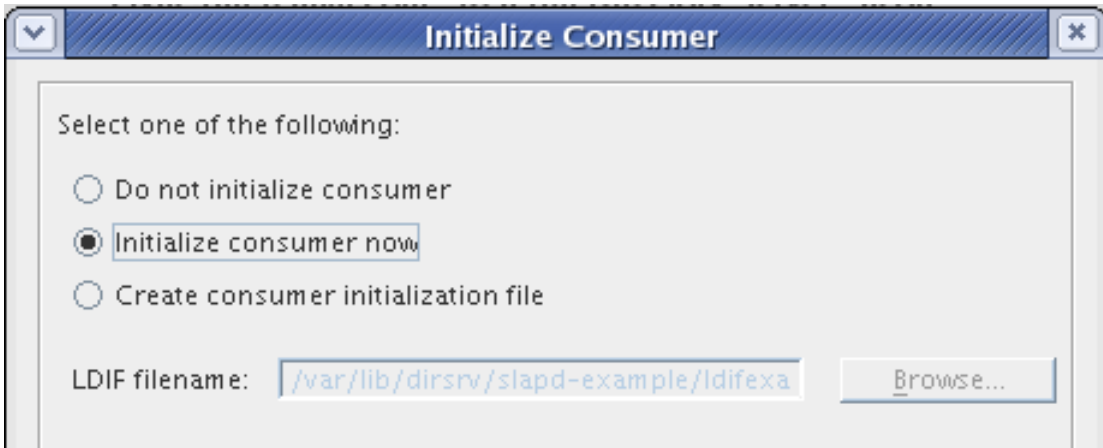
☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Replication will take place between: and

Hit **Next**.

6. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see [Section 10, “Initializing Consumers”](#). For multi-master replication, consider the following:

- Ensure one supplier has the complete set of data to replicate to the other suppliers. Use this one supplier to initialize the replica on all other suppliers in the multi-master replication set.
- Initialize the replicas on the consumer servers from any of the multi-master suppliers.
- Do not try to *reinitialize* the servers when the replication agreements are set up. For example, do not initialize server1 from server2 if server2 has already been initialized from server1. In this case, select **Do not initialize consumer**.



Initialize Consumer

Select one of the following:

☐ Do not initialize consumer

☒ Initialize consumer now

☐ Create consumer initialization file

LDIF filename:

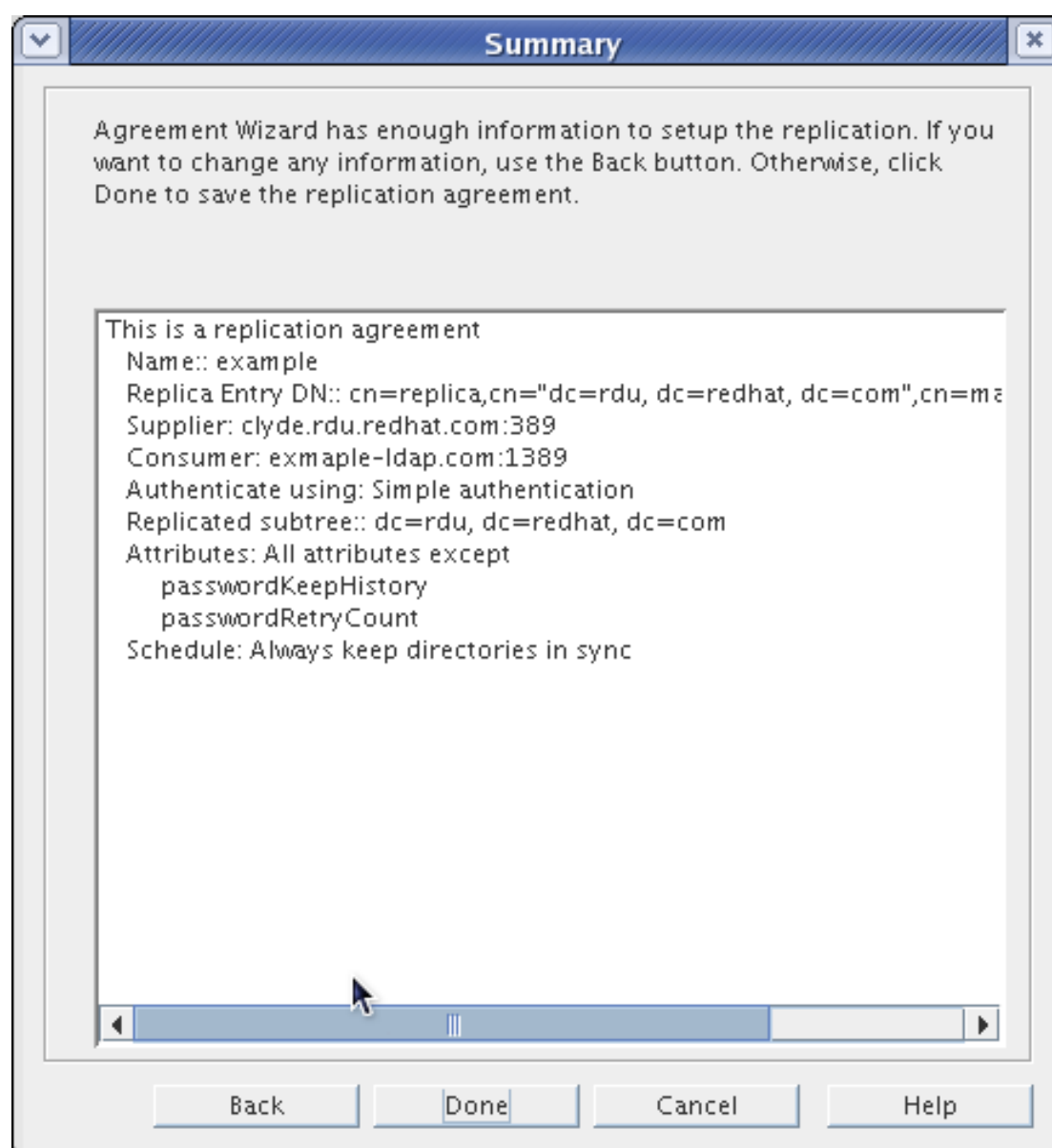


NOTE

Replication *will not* begin until the consumer is initialized.

Hit **Next**.

7. The final screen shows the settings for the replication agreement, as it will be included in the `dse.ldif` file. Hit **Done** to save the agreement.



The replication agreement is set up.

**NOTE**

At the end of this procedure, all supplier servers will have mutual replication agreements, which means that they can accept updates from each other.

**NOTE**

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

5.4. Preventing Monopolization of the Consumer in Multi-Master Replication

One of the features of multi-master replication is that a supplier acquires exclusive access to the consumer for the replicated area. During this time, other suppliers are locked out of direct contact with the consumer. If a supplier attempts to acquire access while locked out, the consumer sends back a busy response, and the supplier sleeps for several seconds before making another attempt. Normally, this is all right; the supplier simply sends its update to another consumer while the first consumer is locked and then send updates when the first consumer is free again.

A problem can arise if the locking supplier is under a heavy update load or has a lot of pending updates in the changelog. If the locking supplier finishes sending updates and then has more pending changes to send, it will immediately attempt to reacquire the consumer and will most likely succeed, since the other suppliers usually will be sleeping. This can cause a single supplier to monopolize a consumer for several hours or longer.

Two attributes address this issue, *nsds5ReplicaBusyWaitTime* and *nsds5ReplicaSessionPauseTime*.

- *nsds5ReplicaBusyWaitTime*. The *nsds5ReplicaBusyWaitTime* attribute sets the amount of time in seconds a supplier should wait after a consumer sends back a busy response before making another attempt to acquire access. The default is 3 seconds.
- *nsds5ReplicaSessionPauseTime*. The *nsds5ReplicaSessionPauseTime* attribute sets the amount of time in seconds a supplier should wait between update sessions. Set this interval so that it is at least one second longer than the interval specified for *nsds5ReplicaBusyWaitTime*. Increase the interval as needed until there is an acceptable distribution of consumer access among the suppliers. The default is 0.

These two attributes may be present in the *nsds5ReplicationAgreement* object class, which is used to describe replication agreements. Set the attributes at any time by using `changetype:modify` with the `replace` operation. The change takes effect for the next update session if one is already in progress.



NOTE

If either attribute is set to a negative value, Directory Server sends the client a message and an `LDAP_UNWILLING_TO_PERFORM` error code.

The two attributes are designed so that the *nsds5ReplicaSessionPauseTime* interval will always be at least one second longer than the interval specified for *nsds5ReplicaBusyWaitTime*. The longer interval gives waiting suppliers a better chance to gain consumer access before the previous supplier can re-access the consumer.

- If either attribute is specified but not both, *nsds5ReplicaSessionPauseTime* is set automatically to 1 second more than *nsds5ReplicaBusyWaitTime*.
- If both attributes are specified, but *nsds5ReplicaSessionPauseTime* is less than or equal to *nsds5ReplicaBusyWaitTime*, *nsds5ReplicaSessionPauseTime* is set automatically to 1 second more than *nsds5ReplicaBusyWaitTime*.

If Directory Server has to automatically reset the value of *nsds5ReplicaSessionPauseTime*, the value is changed internally only. The change is not visible to clients, and it not saved to the configuration file. From an external viewpoint, the attribute value appears as originally set.

Replica busy errors are not logged by default because they are usually benign. To see the errors, turn on the replication error logging, log level 8192. The log levels are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

6. Configuring Cascading Replication

This section provides information on setting up cascading replication. The steps described in this section provide a high-level overview of the procedure, and cross-references to the detailed task descriptions are provided at each step.

Setting up cascading replication, as shown in [Figure 8.4, “Cascading Replication”](#), has three major steps, for each server in the scenario, the supplier on server A, which holds a read-write replica; the consumer/supplier on hub server B, which holds a read-only replica; and the consumer on server C, which holds a read-only replica:

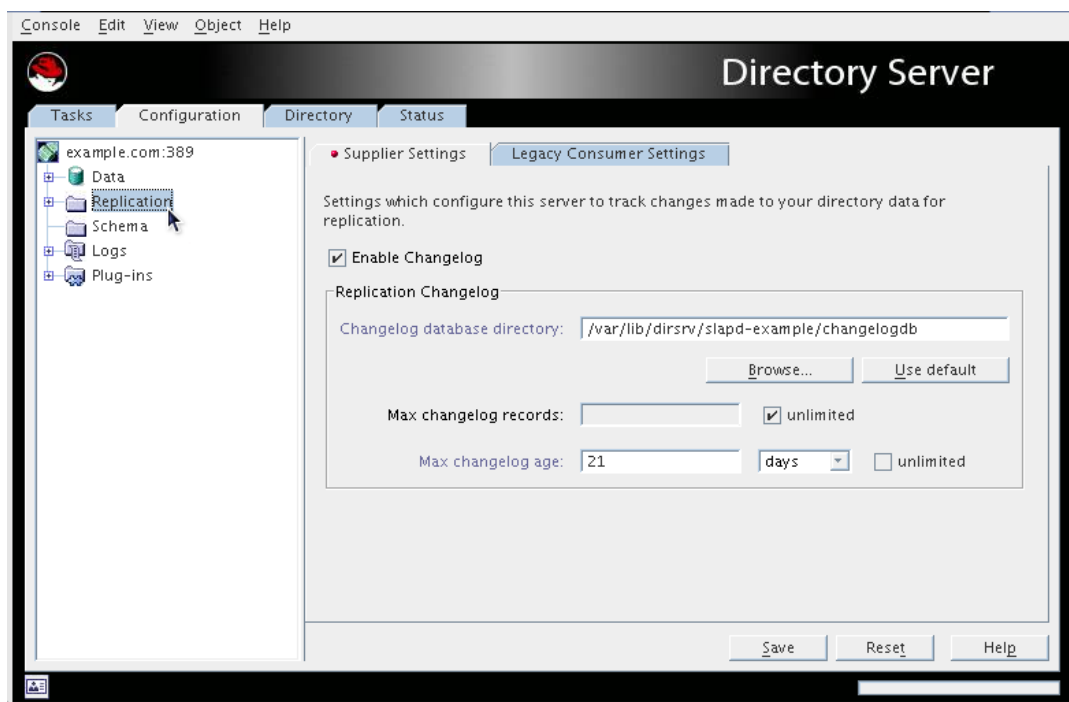
- [Section 6.1, “Configuring the Read-Write Replica on the Supplier Server”](#)

- [Section 6.2, “Configuring the Read-Only Replica on the Consumer Server”](#)
- [Section 6.3, “Configuring the Read-Only Replica on the Hub”](#)
- [Section 6.4, “Setting up the Replication Agreements”](#)

6.1. Configuring the Read-Write Replica on the Supplier Server

Next, configure the supplier server, which holds the original copy of the database:

1. Specify the supplier settings for the server.
 - a. In the Directory Server Console, select the **Configuration** tab.
 - b. In the navigation tree, select the **Replication** folder.
 - c. In the right-hand side of the window, select the **Supplier Settings** tab.



- d. Check the **Enable Changelog** checkbox.
This activates all of the fields in the pane below that were previously grayed out.
- e. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.
- f. Set the changelog parameters for the number and age of the log files.
Clear the unlimited checkboxes to specify different values.

g. Click **Save**.

2. Specify the replication settings required for a read-write replica.

a. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

The **Replica Settings** tab opens in the right-hand side of the window.

b. Check the **Enable Replica** checkbox.

c. In the **Replica Role** section, select the **Single Master** radio button.

d. In the **Common Settings** section, specify a **Replica ID**, which is an integer between 1 and 65534, inclusive.

The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

The purge delay is how often the state information stored in the replicated entries is deleted.

f. Click **Save**.

After setting up the supplier replica, begin configuring the replication agreements.

6.2. Configuring the Read-Only Replica on the Consumer Server

1. Create the database for the read-only replica if it does not exist. See [Section 1.1, “Creating Suffixes”](#) for instructions on creating suffixes.

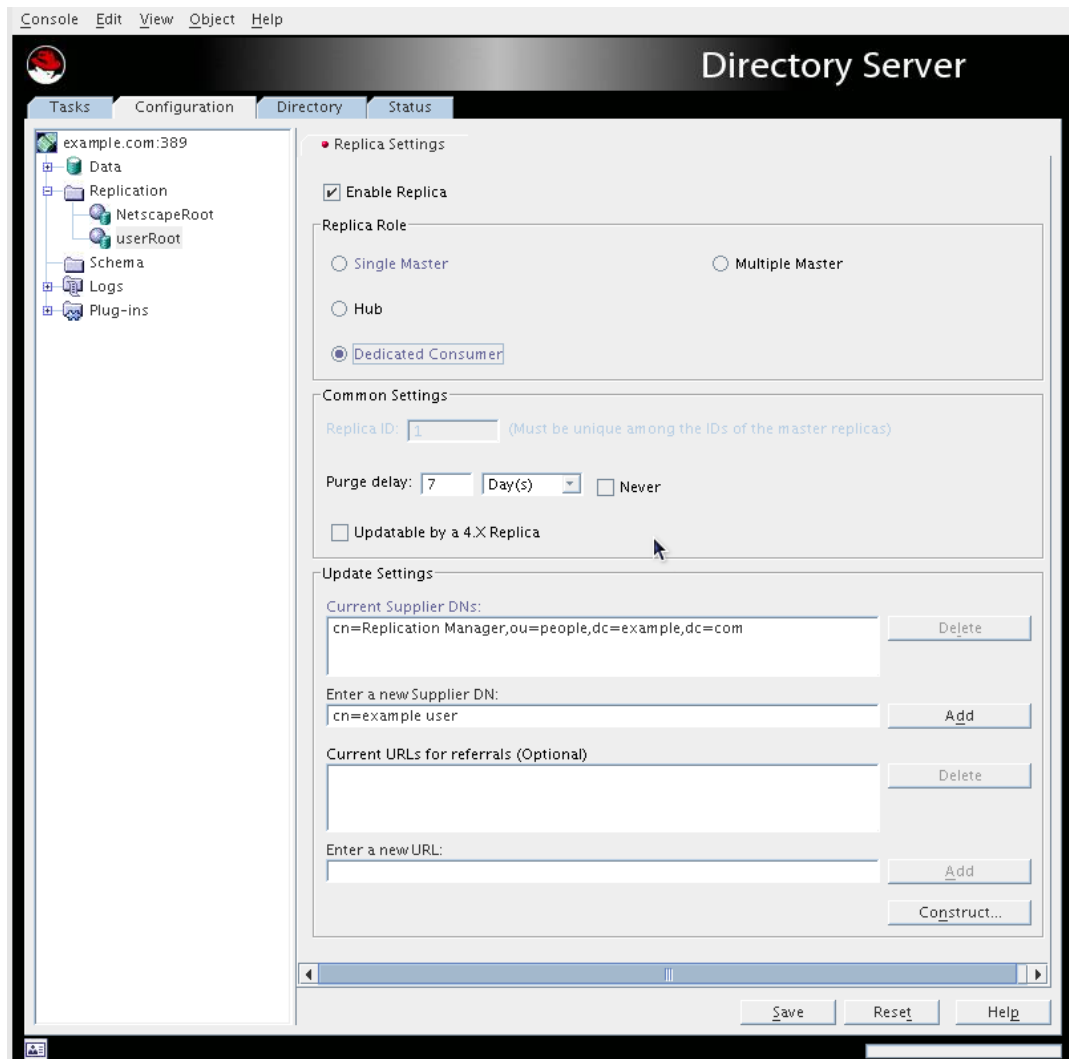
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in [Section 3, “Creating the Supplier Bind DN Entry”](#).

3. Specify the replication settings required for a read-only replica.

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

b. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

The **Replica Settings** tab for that database opens in the right-hand side of the window.



- c. Check the **Enable Replica** checkbox.
- d. In the **Replica Role** section, select the **Dedicated Consumer** radio button.
- e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

This option indicates how often the state information stored in the replicated entries is purged.

- f. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.



NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

- g. Specify the URL for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

In cascading replication, referrals are automatically sent to the hub server, which in turn refers the request to the original supplier. Therefore, set a referral to the original supplier to replace the automatically generated referral.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration, then configure the hub replica.

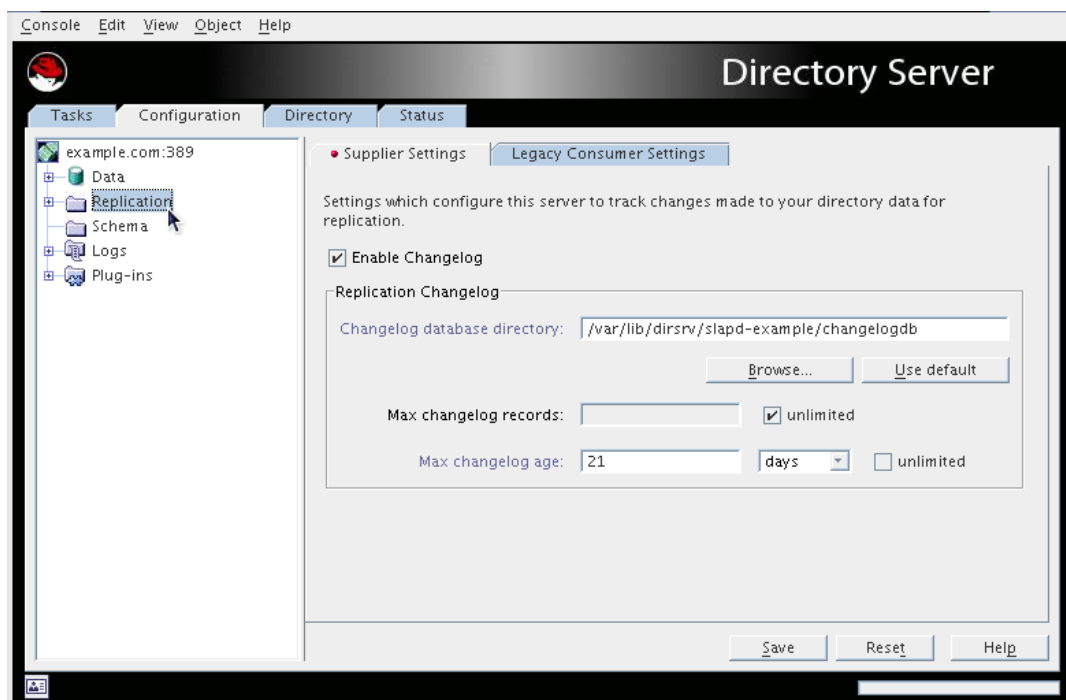
6.3. Configuring the Read-Only Replica on the Hub

Do this to set up a hub, which receives replication updates from the supplier and propagates them to consumers:

1. Create the database for the read-only replica if it does not exist. See [Section 1.1, “Creating Suffixes”](#) for instructions on creating suffixes.
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in [Section 3, “Creating the Supplier Bind DN Entry”](#).
3. Create the changelog for the hub server.

The hub must maintain a changelog even though it does not accept update operations because it records the changes sent from the supplier server.

- a. In the Directory Server Console, select the **Configuration** tab.
- b. In the navigation tree, select the **Replication** folder.
- c. In the right-hand side of the window, select the **Supplier Settings** tab.



- d. Check the **Enable Changelog** checkbox.

This activates all of the fields in the pane below that were previously grayed out.

- e. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.
- f. Set the changelog parameters for the number and age of the log files.

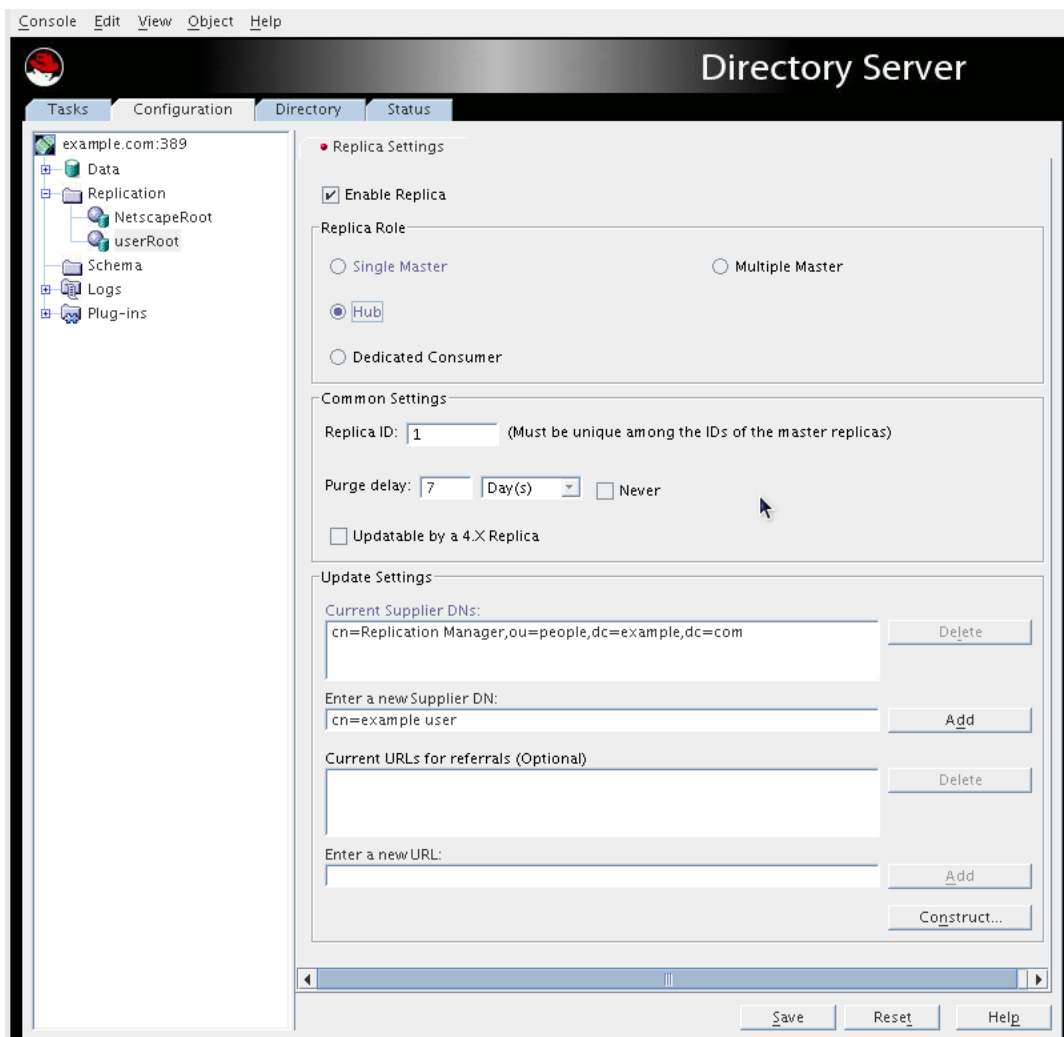
Clear the unlimited checkboxes to specify different values.

- g. Click **Save**.

4. Specify the required hub replica settings.

- a. In the Directory Server Console, select the **Configuration** tab.
- b. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

The **Replica Settings** tab for that database opens in the right-hand side of the window.



- c. Check the **Enable Replica** checkbox.
- d. In the **Replica Role** section, select the **Hub** radio button.
- e. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

This option indicates how often the state information stored in the replicated entries is purged.

- f. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.

The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

**NOTE**

There can be multiple supplier bind DN's per consumer but only one supplier DN per replication agreement.

- g. Specify the URL for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients bind over a regular connection; this has a URL in the form `ldap://hostname:port`. For clients to bind to the supplier using SSL, use this field to specify a referral of the form `ldaps://hostname:port`, where the `s` in `ldaps` indicates a secure connection.

5. Click **Save**.

When all the hubs are configured, then configure the supplier replica.

6.4. Setting up the Replication Agreements

Cascading replication requires two sets of replication agreements, the first between the supplier and the hub and the second between the hub and the consumer. To set up the replication agreements, do the following:

1. Create the replication agreement on the supplier for the hub, then use the supplier server to initialize the replica on the hub server.
2. Then create the replication agreement on the hub for each consumer, and initialize the consumer replicas from the hub.

To set up a replication agreement, do the following:

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.

Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.
3. In the **Source and Destination** screen, fill in the URL for the consumer and the supplier bind

DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.

Source and Destination

Provide server and content information:

Supplier
example.com:389

Consumer
example-ldap.com:1389 Other...

Connection
☐ Using encrypted SSL connection
Authenticate using:
☐ SSL client authentication
☒ Simple authentication
Bind as: cn=Replication Manager,ou=people,dc=example
Password: *****

Subtree:
dc=example, dc=com

Back Next Cancel Help

- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.
- The port listed is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If SSL is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for SSL client authentication. Otherwise, fill in the supplier bind

DN and password.

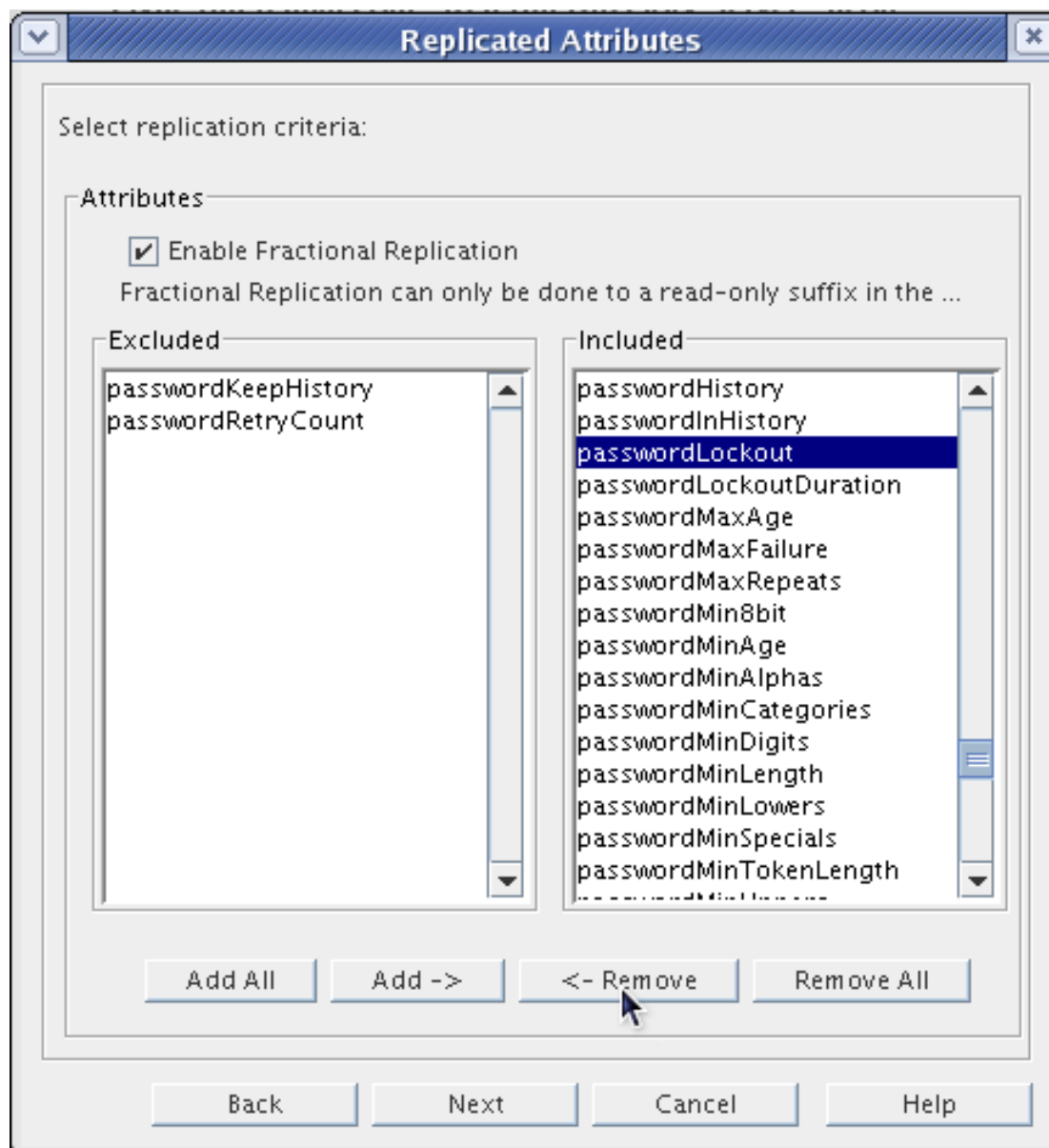


NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

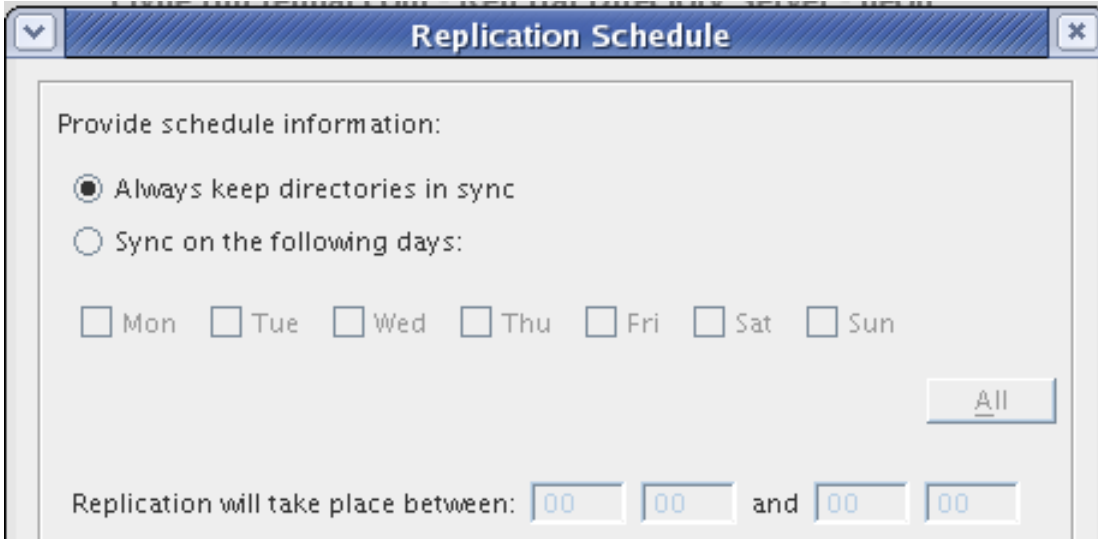
Hit **Next**.

4. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** checkbox. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.

**NOTE**

To safeguard against potential integrity problems, the consumer in fractional replication must be a dedicated consumer, not a multi-master supplier or hub. This is not enforced at the time the replication agreement is made, but replication will fail if the consumer is not a read-only replica.

5. Set the schedule for when replication runs. By default, replication runs continually.



Replication Schedule

Provide schedule information:

☒ Always keep directories in sync

☐ Sync on the following days:

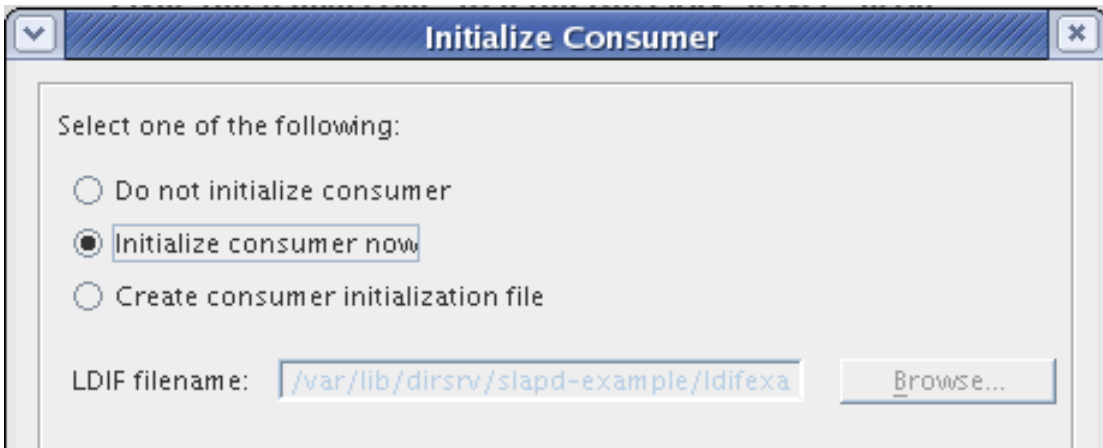
☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Replication will take place between: and

Hit **Next**.

6. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see [Section 10, “Initializing Consumers”](#). For cascading replication, consider the following:

- Create the supplier-hub replication agreement on the supplier first, and initialize the hub from the supplier.
- Create the hub-consumer replication agreements on the hub, and initialize the consumers from the hub.



Initialize Consumer

Select one of the following:

☐ Do not initialize consumer

☒ Initialize consumer now

☐ Create consumer initialization file

LDIF filename:

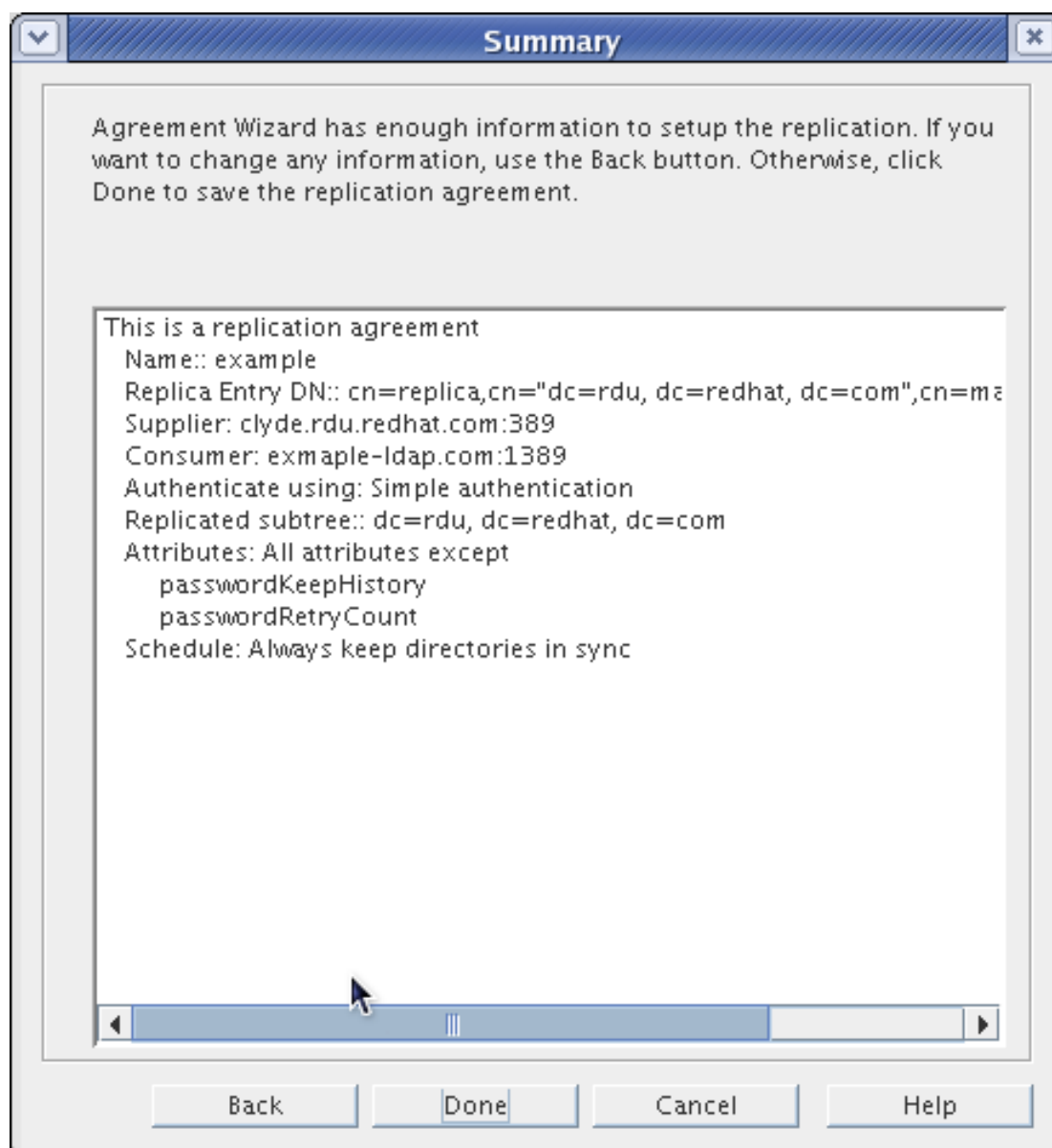


NOTE

Replication *will not* begin until the consumer is initialized.

Hit **Next**.

7. The final screen shows the settings for the replication agreement, as it will be included in the `dse.ldif` file. Hit **Done** to save the agreement.



**NOTE**

After creating a replication agreement, the connection type (SSL or non-SSL) cannot be change because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

7. Configuring Replication from the Command Line

Replication can be configured on the command line by creating the appropriate replica and agreement entries on the servers. The process follows the same order as setting up replication through the Directory Server Console:

1. Create the supplier bind DN on every consumer, hub, and multi-master supplier ([Section 3, “Creating the Supplier Bind DN Entry”](#)).
2. If the corresponding database and suffix do not exist on one of the replicas, create it ([Section 1.1, “Creating Suffixes”](#)).
3. Configure the supplier replicas ([Section 7.1, “Configuring Suppliers from the Command Line”](#)).
4. Configure consumers ([Section 7.2, “Configuring Consumers from the Command Line”](#)).
5. Configure hubs for cascading replication ([Section 7.3, “Configuring Hubs from the Command Line”](#)).
6. Create the replication agreements ([Section 7.4, “Configuring Replication Agreements from the Command Line”](#)). For cascading replication, create the agreement between the supplier and hub, then between the hub and consumers; for multi-master, create the agreements between all suppliers, then between the suppliers and consumers.
7. Lastly, initialize all of the consumers ([Section 7.5, “Initializing Consumers Online from the Command Line”](#)), if the consumers were not initialized when the replication agreement was created.

7.1. Configuring Suppliers from the Command Line

There are two steps to setting up the supplier replica. First, the changelog must be enabled, which allows the supplier to track changes to the Directory Server. Then, the supplier replica is created.

1. On the supplier server, use `ldapmodify` to create the `changelog`¹ entry.

```
ldapmodify -v -h supplier1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb
```

There is one important attribute with the changelog, *nsslapd-changelogdir*, which sets the directory where the changelog is kept.

The changelog entry attributes are described in [Table 8.1, “Changelog Attributes”](#). These attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

2. Create the supplier replica.

```
ldapmodify -v -h supplier1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=replica,cn="dc=example,dc=com",cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 7
nsds5replicatype: 3
nsds5flags: 1
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=replication manager,cn=config
```

- *nsds5replicaroot* sets the subtree (suffix) which is being replicated.
- *nsds5replicatype* sets what kind of replica this database is. For either a single master or a multi-master supplier, this value must be 3.
- *nsds5replicaid* sets the replica ID. The value must be unique among all suppliers and hubs; the valid range is 1 to 65534.
- *nsds5ReplicaPurgeDelay* sets how long the supplier holds onto its change record before deleting it. The default value is 604800 (one week).
- *nsds5flags* sets whether the replica writes to the changelog. For a supplier, this value must

¹ The file location here is the default location for Red Hat Enterprise Linux 5 i386. For the default location on other platforms, see [Section 1, “Directory Server File Locations”](#).

be 1.

The replica entry attributes are described in [Table 8.2, “Replica Attributes”](#).

After creating every supplier which will take part in the replication setup, then begin creating the replication agreements.

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: extensibleObject	An object class which allows any other object class or attribute to be added to an entry.	
cn: changelog5	The naming attribute for the changelog entry.	Any string; the default usage is to set the common name to changelog5.
nsslapd-changelogdir: <i>directory</i>	Sets the file and directory changelog, to which the Directory Server writes changes.	Any directory; the default is <code>/var/lib/dirsrv/slapd-<i>instance_name</i>/changelog</code>

Table 8.1. Changelog Attributes

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: extensibleObject	An object class which allows any other object class or attribute to be added to an entry.	
objectclass: nsds5replica	An object class which allows replication attributes to be added to an entry.	
cn: replica	The naming attribute for the replica.	Any string; the default usage is to set the common name to <code>replica</code> for every configured replica.
nsds5replicaroot: <i>suffix</i>	Sets which subtree is replicated.	A root suffix associated with a database, since the entire database is replicated. For example:

Object Class or Attribute	Description	Values
		dc=example,dc=com
nsds5replicaid: <i>number</i>	The ID of the replica. This must not be set for consumers or hubs, but is required for suppliers.	1 to 65534, inclusive.
nsds5replicatype: <i>number</i>	Sets the type of replica, either read-only or read-write.	2 for consumers and hubs (read-only replicas) 3 for both single and multi-master suppliers (read-write replicas)
nsds5flags: <i>number</i>	Sets whether the replica writes to the changelog.	0 means the replica does not write to the changelog; this is the default for consumers. 1 means the replica writes to the changelog; this is the default for hubs and suppliers.
nsds5ReplicaPurgeDelay: <i>number</i>	Sets the period of time in seconds to wait before purging the entries from the changelog. This is not required for consumers, but is required for hubs and suppliers.	0 (keep forever) to 2147483647 (the maximum 32-bit integer); the default value is 604800, one week.
nsds5ReplicaBindDN: <i>DN</i>	The supplier bind DN used by the supplier to bind to the consumer. This is required for consumers, hubs, and multi-master suppliers, but not for single-master suppliers.	Any DN; the recommended DN is cn=ReplicationManager,cn=config.

**NOTE**

For security, it is strongly recommended that you do *not* use the Directory Manager as the supplier

Object Class or Attribute	Description	Values
		bind DN.
<code>nsds5replicareferral</code> : <i>URL</i>	<i>Optional.</i> An LDAP URL which a consumer or hub to which a consumer or hub can forward update requests. By default, update requests are sent to the masters for the consumer; use this parameter to override the default.	Any LDAP URL. For example: <pre>nsds5replicareferral: ldap://supplier1.example.com:389</pre>

Table 8.2. Replica Attributes

7.2. Configuring Consumers from the Command Line

On the consumer host, such as `consumer1.example.com`, create the replica entry. This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is. There are four key attributes:

- `nsds5replicaroot` sets the subtree (suffix) which is being replicated.
- `nsds5replicatype` sets what kind of replica this database is. For a consumer, this value must be 2.
- `nsds5ReplicaBindDN` give the DN as which the supplier will bind to the consumer to make changes.
- `nsds5flags` sets whether the replica writes to the changelog. For a consumer, this value must be 0.

This `ldapmodify` creates a new consumer replica on the `consumer1.example.com` host for the `dc=example,dc=com` subtree.

```
ldapmodify -v -h consumer1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=replica,cn="dc=example,dc=com",cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
```

```
nsds5replicatype: 2
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5flags: 0
```

The replica entry attributes are described in [Table 8.2, “Replica Attributes”](#). These attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

7.3. Configuring Hubs from the Command Line

Hubs are intermediate read-only replicas which receive updates from suppliers and pass them on to other consumers. These are part of the cascading replication scenario, described in [Section 2.3, “Cascading Replication”](#). Creating the hub has two steps: first, creating the changelog database since the hub keeps a record of changes sent by the supplier, and second, configuring the hub replica.

1. On the hub server, such as `hub1.example.com`, use `ldapmodify` to create the changelog¹ entry.

```
ldapmodify -v -h hub1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb
```

There is one important attribute with the changelog, `nsslapd-changelogdir`, which sets the directory where the changelog is kept.

The changelog entry attributes are described in [Table 8.1, “Changelog Attributes”](#). These attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

2. On the hub host, create the replica entry. This `ldapmodify` command creates a new hub replica on the `hub1.example.com` host for the `dc=example,dc=com` subtree.

```
ldapmodify -v -h hub1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=replica,cn="dc=example,dc=com",cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
```



```
nsds5replicatype: 2
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5flags: 1
```

This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is. There are five key attributes:

- *nsds5replicaroot* sets the subtree (suffix) which is being replicated.
- *nsds5replicatype* sets what kind of replica this database is. For a hub, this value must be 2.
- *nsds5ReplicaPurgeDelay* sets how long the hub holds onto its change record before deleting it. The default value is 604800 (one week).
- *nsds5ReplicaBindDN* give the DN as which the supplier will bind to the hub to make changes.
- *nsds5flags* sets whether the replica writes to the changelog. For a hub, this value must be 1.

The replica entry attributes are described in [Table 8.2, “Replica Attributes”](#). These attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

7.4. Configuring Replication Agreements from the Command Line

When setting up replication agreements, first set them up between all suppliers, then between the suppliers and the hubs, and last between the hub and the consumers.

The replication agreement has to define seven things:

- The consumer host (*nsds5replicahost*) and port (*nsds5replicaport*).
- The DN for the supplier to use to bind with the consumer (*nsds5ReplicaBindDN*), the way that the supplier binds (*nsds5replicabindmethod*), and any credentials required (*nsds5replicabindcredentials*).
- The subtree being replicated (*nsds5replicaroot*).
- The replication schedule (*nsds5replicaupdateschedule*).
- Any attributes which will *not* be replicated (*nsds5replicatedattributelist*).

Use `ldapmodify` to add a replication agreement to every supplier for every consumer which it will updated. For example:

```
dn: cn=ExampleAgreement,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replicationagreement
cn: ExampleAgreement
nsds5replicahost: consumer1
nsds5replicaport: 389
nsds5ReplicaBindDN: cn=replication manager
nsds5replicabindmethod: SIMPLE
nsds5replicaroot: dc=example,dc=com
description: agreement between supplier1 and consumer1
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE
authorityRevocationList
nsds5replicacredentials: {DES}UXRbhvozeN9LWdueOEbPeQ==
nsds5BeginReplicaRefresh: start
```

The replication agreement attributes are listed in [Table 8.3, “Replication Agreement Attributes”](#). These attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

After creating every replication agreement, begin initializing consumers.

Object Class or Attribute	Description	Values
objectclass: top	Required object class for every entry.	
objectclass: nsds5replicationagreement	An operational object class which contains the replication agreement attributes.	
cn: <i>agreement_name</i>	The naming attribute for the replication agreement.	Any string.
nsds5replicahost: <i>hostname</i>	Gives the hostname of the consumer server; the hostname can be the fully qualified host and domain name. If TLS/SSL is enabled, the fully-qualified domain name is required.	Any hostname. For example: <pre>nsds5replicahost: consumer1</pre>
nsds5replicaport: <i>number</i>	Gives the LDAP port for the consumer server. To use TLS/SSL, give the secure port number (636 by default) and set the <i>nsds5ReplicaTransportInfo</i> attribute to SSL.	Any port number.
nsds5replicatransportinfo:	To use TLS/SSL, set this	SSL

Object Class or Attribute	Description	Values
<i>method</i>	parameter to <i>SSL</i> . If TLS/SSL is not used, this attribute can be absent.	
nsds5ReplicaBindDN: <i>DN</i>	The supplier bind DN used by the supplier to bind to the consumer. This is required for consumers, hubs, and multi-master suppliers, but not for single-master suppliers.	Any DN; the recommended DN is <code>cn=ReplicationManager,cn=config</code> .
nsds5replicabindmethod: <i>type</i>	The connection type for replication between the servers.	<code>SIMPLE</code> or <code>SSLCLIENTAUTH</code>
nsds5replicabindcredentials: <i>hash</i>	<i>Only for simple authentication.</i> Stores the hashed password used with the bind DN given for simple authentication.	
nsds5replicaroot: <i>suffix</i>	Sets which subtree is replicated.	A root suffix associated with a database, since the entire database is replicated. For example: <code>dc=example,dc=com</code>
description: <i>text</i>	A text description of the replication agreement.	Any text string. It is advisable to make this a useful description, such as <i>agreement between supplier1 and consumer1</i> .
nsds5replicatedattributelist: <i>'(objectclass=*)' \$ EXCLUDE attributes</i>	<i>Optional.</i> Sets which attributes will <i>not</i> be replicated. The filter must be set to <code>"(objectclass=*)"</code> , and the list of attributes are separated by a single space.	<code>'(objectclass=*)' \$ EXCLUDE userPassword manager cn</code>
nsds5replicaupdateschedule: <i>start_time end_time days</i>	Sets the start and end time for the replication updates and the days on which replication occurs. If the schedule is omitted, replication will take place all the time.	Has the following value, with the start (SSSS) and end (EEEE) times set in the form HHMM The times are given in 24

Object Class or Attribute	Description	Values
		<p>hour clock format, so 0000 is midnight and 2359 is 11:59 PM. For example, the setting 1030 1630 schedules replication from 10:30 AM to 4:30 PM. The times cannot wrap around midnight, so the setting 2300 0100 is not valid.</p> <p>The days ranging from 0 (Sunday) to 6 (Saturday). Setting 06 schedules replication on Sunday and Saturday, while 135 schedules replication on Monday, Wednesday, and Friday.</p> <pre>nsds5replicaupdateschedule: SSSS FFFF DDDDDDD</pre> <p>For example, this schedules replication between midnight (0000) and 5am (0500) on Monday and Tuesday:</p> <pre>nsds5replicaupdateschedule: 0000 0500 12</pre>
nsds5BeginReplicaRefresh: start	<p><i>Optional.</i> Performs an online (immediate) initialization of the consumer. If this is set, the attribute is only present as long as the consumer is being initialized; when the initialization is complete, the attribute is deleted automatically.</p> <p>If this is not set, then consumer initialization must be performed manually.</p>	<p>start</p> <p>To initialize the consumer, this attribute must have a value of <i>start</i>; any other value is ignored.</p>

Table 8.3. Replication Agreement Attributes

7.5. Initializing Consumers Online from the Command Line

An online initialization can be initiated from the command line by adding the `nsds5replicarefresh` attribute to the replication agreement entry. If the attribute is included when the replication agreement is created, initialization begins immediately. It can be added later to initialize the consumer at any time. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

```
ldapsearch -h supplier1.example.com -p 389 -D "cn=directory manager" -w
password -s sub
        -b cn=config "(objectclass=nsds5ReplicationAgreement)"
```

This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the `nsds5BeginReplicaRefresh` attribute:

```
ldapmodify -h supplier1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=ExampleAgreement,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
replace: nsds5beginreplicarefresh
nsds5beginreplicarefresh: start
```

`ldapmodify` does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the `ldapmodify` utility by hitting **Ctrl+C**.

When the initialization is complete, the `nsds5beginreplicarefresh` attribute is automatically deleted from the replication agreement entry.



NOTE

Initializing consumers from the command line is also explained in [Section 10.3, “Initializing Consumers Online Using the Command Line”](#). Manually initializing consumers is explained in [Section 10.4, “Manual Consumer Initialization Using the Command Line”](#). The replication monitoring attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

To keep data integrity, initialize the consumer databases from the appropriate supplier.

Depending on the replication scenario, this can be more difficult in mixed replication environments, but, even when manually initializing consumers, consider four things:

- Use one supplier, a *data master*, as the source for initializing consumers.
- Do not *reinitialize* a data master when the replication agreements are created. For example, do not initialize server1 from server2 if server2 has already been initialized from server1.
- For a multi-master scenario, initialize all of the other master servers in the configuration from one master.
- For cascading replication, initialize all of the hubs from a supplier, then initialize the consumers from the hubs.

8. Making a Replica Updatable

Making a read-only server writable means changing the replica from a dedicated consumer or a hub to a supplier.

1. Make sure there are no updates in progress.
2. Stop the supplier server.
3. Open the Directory Server Console for the read-only replica.
4. In the **Configuration** tab, select **Replication**. In the right pane, select the **Enable changelog** checkbox.
5. Select the suffix, and, in the **Replica Settings** tab, change the replica role to a single master or multi-master, and assign a unique replica ID.
6. Save the changes, and restart the server.

9. Deleting the Changelog

The changelog is a record of all modifications on a given replica that the supplier uses to replay these modifications to replicas on consumer servers (or suppliers in the case of multi-master replication). 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. After deleting the changelog, reinitialize the consumers and begin the replication process afresh. To delete the changelog, either remove it or move it to a new location.

- [Section 9.1, “Removing the Changelog”](#)

- [Section 9.2, “Moving the Changelog to a New Location”](#)

9.1. Removing the Changelog

To remove the changelog from the supplier server, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. Select the **Replication Agreements** folder in the left navigation tree and then the **Supplier Server Settings** tab in the right pane.
3. Clear the **Enable Changelog** checkbox.

This deletes the changelog.
4. Click **Save**.
5. Restart the Directory Server.
6. Reinitialize the consumers, as in [Section 10, “Initializing Consumers”](#).



NOTE

If the changelog is removed, the consumer servers must be reinitialized.

9.2. Moving the Changelog to a New Location

To delete the changelog while the server is still running and continuing to log changes, simply move the changelog to a new location. By moving the changelog, a new changelog is created in the specified directory, and the old changelog is deleted. Changing the location of the changelog requires consumer reinitialization.

10. Initializing Consumers

Once a replication agreement is created, the consumer must be *initialized*; that is, the data must be physically copied from the supplier server to the consumer servers. This section first describes consumer initialization in detail and then provides instructions on the two different methods for initializing consumers.



NOTE

Replication *will not* begin until the consumer is initialized.

- [Section 10.1, “When to Initialize a Consumer”](#)
- [Section 10.2, “Online Consumer Initialization Using the Console”](#)
- [Section 10.3, “Initializing Consumers Online Using the Command Line”](#)
- [Section 10.4, “Manual Consumer Initialization Using the Command Line”](#)
- [Section 10.5, “Filesystem Replica Initialization”](#)

10.1. When to Initialize a Consumer

Consumer initialization involves copying data from the supplier server to the consumer server. Once the subtree has been physically placed on the consumer, the supplier server can begin replaying update operations to the consumer server.

Under normal operations, the consumer should not ever have to be reinitialized. However, any time there is a chance that there is a big discrepancy between the supplier's data and the consumer's, reinitialize the consumer. For example, if the data on the supplier server is restored from backup, then all consumers supplied by that server should be reinitialize. As another example, if the supplier has not been able to contact the consumer for a long time, like a week, the supplier may determine that the consumer is too far out of date to be updated, and must be reinitialized.

The consumer can either be initialized online using the Console or manually using the command-line. Online consumer initialization using the Console is an effective method of initializing a small number of consumers. However, since each replica is initialized in sequence, this method is not suited to initializing a large number of replicas. Online consumer initialization is the method to use when the consumer is initialized as part of configuring the replication agreement on the supplier server.

Manual consumer initialization using the command-line is a more effective method of initializing a large number of consumers from a single LDIF file.

10.2. Online Consumer Initialization Using the Console

Online consumer initialization using the Console is the easiest way to initialize or reinitialize a consumer. However, for replicating across a slow link, this process can be very time-consuming, and manual consumer initialization using the command-line may be a more efficient approach. This is described in more detail [Section 10.4, “Manual Consumer Initialization Using the Command Line”](#).



NOTE

When a consumer server is being initialized using the online consumer creation method, all operations (including searches) on the replica are referred to the supplier server until the initialization process is completed.

To initialize or reinitialize a consumer online, do the following:

1. Create a replication agreement.
2. On the supplier server, on the Directory Server Console, select the **Configuration** tab.
3. Expand the **Replication** folder, then expand the replicated database. Right-click the replication agreement, and choose **Initialize Consumer** from the pop-up menu.

A message opens warning that any information already stored in the replica on the consumer will be removed.

4. Click **Yes** in the confirmation box.

Online consumer initialization begins immediately. To check the status of the online consumer initialization, open the **Summary** tab in the **Status** box. If online consumer initialization is in progress, the status shows that a replica is being initialized.

To update this window, right-click the replicated database icon in the navigation tree, and choose **Refresh Replication Agreements**. When online consumer initialization finishes, the status changes to reflect this.

For more information about monitoring replication and initialization status, see [Section 17, "Monitoring Replication Status"](#).

10.3. Initializing Consumers Online Using the Command Line

Online consumer initialization can be performed through the command line by adding the `nsds5BeginReplicaRefresh` attribute to the replication agreement entry. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

```
ldapsearch -h supplier1.example.com -p 389 -D "cn=directory manager" -w  
password -s sub  
-b cn=config "(objectclass=nsds5ReplicationAgreement)"
```

This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the `nsds5BeginReplicaRefresh` attribute:

```
ldapmodify -h supplier1.example.com -p 389 -D "cn=directory manager" -w  
password
```

```
dn: cn=ExampleAgreement,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
replace: nsds5beginreplicarefresh
nsds5beginreplicarefresh: start
```

`ldapmodify` does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the `ldapmodify` utility by hitting **Ctrl+C**.

To check the initialization status, do an `ldapsearch` for the replication agreement entry.

```
ldapsearch -h hostname -p port -D "cn=directory manager" -w password -s base
-b 'cn=ExampleAgreement,cn="dc=example,dc=com", cn=mapping tree,
cn=config' '(objectclass=*)'
```

If the `nsds5BeginReplicaRefresh` attribute is present, the initialization is still in progress. If the initialization is complete, then the attribute `nsds5ReplicaLastInitStatus` shows the status. If the initialization was successful, the value of `nsds5ReplicaLastInitStatus` is `Total update succeeded`. If the initialization was not successful, this attribute shows information about the error; check the error logs for both the supplier and consumer for additional information.

The replication monitoring attributes are described in more detail in the *Directory Server Configuration, Command, and File Reference*.

10.4. Manual Consumer Initialization Using the Command Line

Manual consumer initialization using the command-line is the fastest method of consumer initialization for sites that are replicating very large numbers of entries. However, the manual consumer initialization process is more complex than the online consumer initialization process. Red Hat suggests using the manual process whenever the online process is inappropriate due to performance concerns.

Initializing or reinitializing a server manually has three steps:

1. Create a replication agreement.
2. Export the replica on the supplier server to an LDIF file.

See [Section 10.4.1, “Exporting a Replica to LDIF”](#).

3. Import the LDIF file with the supplier replica contents to the consumer server.

See [Section 10.4.2, “Importing the LDIF File to the Consumer Server”](#).

10.4.1. Exporting a Replica to LDIF

There are three ways to convert a replica database to LDIF:

- When creating a replication agreement, by selecting **Create consumer initialization file** in the **Initialize Consumer** dialog box of the **Replication Agreement Wizard**.
- From the Directory Server Console, by right-clicking the replication agreement under the **Replication** folder and choosing **Create LDIF File** from the pop-up menu.
- From the command-line by using the export command, as described in [Section 2.3, “Exporting to LDIF from the Command-Line”](#).

10.4.2. Importing the LDIF File to the Consumer Server

Import the LDIF file which contains the supplier replica contents to the consumer server by using the import features in the Directory Server Console or by using either the `ldif2db` script or `ldif2db.pl` script. Both import methods are described in [Section 1.3, “Importing from the Command-Line”](#).



NOTE

With the `ldif2db.pl` script, the LDIF file import operation does not require a server restart. For more information on command-line scripts, see the *Directory Server Configuration, Command, and File Reference*.

10.5. Filesystem Replica Initialization

A very large database, such as one with several million entries, can take an hour or more to initialize a consumer from the Console or even with manual initialization. To save time, use *filesystem replica initialization*.

Directory Server has the capability to initialize a replica using the database files from the supplier server. This avoids the need to rebuild the consumer database and can be done at essentially the speed of the network between the two servers by transferring the files with FTP or NFS, for example. Instead of sending entries via LDAP to replica servers, filesystem replica initialization populates the new database on the destination server by *backing up* the supplier database on one server and *restoring* the database on the destination server.

This method of initializing consumers is especially useful in replication over wide-area networks or over networks with slow or unstable connections.

For smaller databases, Red Hat recommends using manual initialization or initialize consumers from the Console.



NOTE

The destination server must have the same architecture and the same bit size as the supplier server for the initialization to succeed. For example, Red Hat Enterprise Linux 32-bit to Red Hat Enterprise Linux 32-bit.

10.5.1. Initializing the Consumer Replica from the Backup Files

1. Create a new database on the destination server to match the database from the source server.

Before initializing the consumer from the backup files, be certain that the appropriate database has been created on the destination server so that the database exists to be restored and initialized.

2. Enable replication on the backend as a dedicated consumer.
3. If there is already a replication agreement to that host and port, then replication should resume immediately after running the restore script. Otherwise, create the replication agreement on the source server (or whatever server is the supplier), and select the **Do not initialize consumers at this time** radio button.
4. Stop the source Directory Server if it is running. For example:

```
service dirsrv stop slapd-example
```

5. From the command-line, run the `db2bak` utility, and archive the current directory installation.

```
/usr/lib/dirsrv/slapd-instance_name/db2bak
```

In addition, a new backup can be created by hitting the **Back Up Directory Server** button in the **Tasks** tab of the Directory Server Console and then putting the name of the archive directory in the **Directory:...** field. Alternatively, select any previous back-up to initialize the consumer.

This backup information is recovered in the destination replica.

6. Restart the source Directory Server. For example:

```
service dirsrv start slapd-example
```

7. Copy the archived files to a directory on the destination server, such as `archiveDirectory`.

8. Stop the destination Directory Server if it is running.

```
service dirsrv stop slapd-example2
```

9. On the destination server, restore the archives with the `bak2db` script, using the optional `-n` parameter to specify the backend instance name. This `-n` parameter is similar to the `-n` used with `ldif2db` and `db2ldif`. For example:

```
/usr/lib/dirsrv/slapd-example2/bak2db /tmp/archiveDirectory -n userRoot
```

10. Restart the destination Directory Server. For example:

```
service dirsrv start slapd-example2
```

Replication will begin on schedule as soon as the destination server is restarted.

For more information on using these scripts, see the *Directory Server Configuration, Command, and File Reference*.

11. Forcing Replication Updates

When a Directory Server involved in replication is stopped for regular maintenance, it must be updated immediately when it comes back online. In the case of a supplier in a multi-master environment, the directory information needs to be updated by the other supplier in the multi-master set. In other cases, when a hub or a dedicated consumer is taken offline for maintenance, when they come back online, they need to be updated by the supplier server.

Even if the replication agreements are configured to keep the supplier and consumer servers always in sync, it is not sufficient to bring back up-to-date a server that has been offline for over five minutes. The **Always Keep in Sync** option means that the server generates a replication operation for every update operation it processes. However, if this replication operation cannot be performed because the consumer is offline, the operation times out after 10 minutes.



NOTE

The procedures described in this section can only be used when replication is already set up and consumers have been initialized.

To ensure that directory information will be synchronized immediately when a server comes back online, use either the Directory Server Console on the supplier server that holds the reference copy of the directory information or a customizable script.

11.1. Forcing Replication Updates from the Console

To ensure that replication updates are sent immediately when a consumer or a supplier in a multi-master replication configuration comes back online after a period of time, do the following on the supplier server that holds the most recent version of the directory information:

1. In the Directory Server Console, click the **Configuration** tab, expand the **Replication** folder and database nodes, and select the replication agreement corresponding to the replica to update.
2. Right click the replication agreement, and choose **Send Updates Now** from the drop-down list.

This initiates replication toward the server that holds the information that needs to be updated.

11.2. Forcing Replication Updates from the Command-Line

From the consumer that requires updating, run a script that prompts the supplier to send replication updates immediately. This script is shown in [Example 8.1, “Replicate_Now Script Example”](#).

Copy this example script and name it something like `replicate_now.sh`. Substitute the actual values for the variables listed in [Example 8.1, “Replicate_Now Script Example”](#).



NOTE

This script must be run manually since it cannot be configured to run automatically as soon as the server, which was offline, comes back online again.

```
#!/bin/sh
SUP_HOST=supplier_hostname
SUP_PORT=supplier_portnumber
SUP_MGRDN=supplier_directoryManager
SUP_MGRPW=supplier_directoryManager_password
MY_HOST=consumer_hostname
MY_PORT=consumer_portnumber

ldapsearch -1 -T -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \
-w ${SUP_MGRPW} -b "cn=mapping tree, cn=config"
\"(&(objectclass=nsds5replicationagreement)(nsDS5ReplicaHost=${MY_HOST})
\"(nsDS5ReplicaPort=${MY_PORT}))\" dn nsds5ReplicaUpdateSchedule >
/tmp/$$

cat /tmp/$$ |awk 'BEGIN { s = 0 }/^dn: / { print $0;print "changetype:
modify";print
    "replace: nsds5ReplicaUpdateSchedule";print
    "nsds5ReplicaUpdateSchedule: 0000-2359
    0123456";print "-";print "";print $0;print "changetype: modify";
    print "replace:nsds5ReplicaUpdateSchedule";}
```

```

/^nsds5ReplicaUpdateSchedule: / { s = 1; print $0; }/^$/{if ( $s == 1 ){
print "-" ;
    print ""; }else{ print "nsds5ReplicaUpdateSchedule: 0000-2359
0123456";print "-" ;
    print ""; };s = 0; }

' > /tmp/ldif.$$echo "Ldif is in /tmp/ldif.$$"echo

ldapmodify -c -h ${SUP_HOST} -p ${SUP_PORT} -D "${SUP_MGRDN}" \-w
${SUP_MGRPW}
    -f /tmp/ldif.$$

```

Example 8.1. Replicate_Now Script Example

Variable	Definition
<i>supplier_hostname</i>	Hostname of the supplier to contact for information on replication agreements with the current consumer.
<i>supplier_portnumber</i>	LDAP port in use on the supplier.
<i>supplier_directoryManager</i>	DN of the privileged Directory Manager user on the supplier.
<i>supplier_directoryManager_password</i>	Password of the privileged Directory Manager user on the supplier.
<i>consumer_hostname</i>	Hostname of the current consumer.
<i>consumer_portnumber</i>	LDAP port in use on the consumer.

Table 8.4. Replicate_Now Variables

For the update operation to occur over an SSL connection, modify the `ldapmodify` command in the script with the appropriate parameters and values. For more information on the `ldapmodify` command, see [Section 2, “Managing Entries from the Command-Line”](#) and the *Directory Server Configuration, Command, and File Reference*.

12. 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 master.

By default, three password policy attributes are not replicated, even if other password attributes are. These attributes are related to login failures and lockout periods:

- *passwordRetryCount*
- *retryCountResetTime*
- *accountUnlockTime*

To enable these attributes to be replicated, change the *passwordIsGlobalPolicy* configuration attribute:

```
ldapmodify -h consumer1.example.com -p 389 -D "cn=directory manager" -w password

dn: cn=config
changetype: modify
replace: passwordIsGlobalPolicy
passwordIsGlobalPolicy: 1
```

Changing that value to 1 allows the *passwordRetryCount*, *retryCountResetTime*, and *accountUnlockTime* to be replicated. No other configuration is necessary for the attributes to be included with the replicated attributes.

13. Replication over SSL

The Directory Servers involved in replication can be configured so that all replication operations occur over an SSL connection. To use replication over SSL, first do the following:

- Configure both the supplier and consumer servers to use SSL.
- Configure the consumer server to recognize the supplier server's certificate as the supplier DN. Do this only to use SSL client authentication rather than simple authentication.

These procedures are described in [Chapter 11, Managing SSL](#).

If attribute encryption is enabled, a secure connection is required for replication.



NOTE

Replication configured over SSL with certificate-based authentication will fail if the supplier's certificate is only capable of behaving as a server certificate, and

not also a client during an SSL handshake. Replication with certificate-based authentication uses the Directory Server's server certificate for authentication to the remote server.

When the servers are configured to use SSL, configure an SSL connection for replication in the **Replication Agreement Wizard**. The **Source and Destination** sets how to bind between the supplier and the consumer, and this is where SSL is set.

There are two ways to use SSL for replication:

- Select **SSL Client Authentication**.

With SSL client authentication, the supplier and consumer servers use certificates to authenticate to each other.

- Select **Simple Authentication**.

With simple authentication, the supplier and consumer servers use a bind DN and password to authenticate to each other, which are supplied in the **Replication Agreement Wizard** text fields provided. Simple authentication takes place over a secure channel but without certificates.

Once a replication agreement is created, the connection type (SSL or non SSL) cannot be changed in the agreement because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

Also, the port listed for the consumer is the non-SSL port, even if the Directory Server instance is configured to run over SSL. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.

14. Replicating o=NetscapeRoot for Administration Server Failover

Replication usually occurs between Directory Server user databases to distribute directory data, but it is also possible to use replication to provide failover support for the Administration Server database, o=NetscapeRoot.

1. Install and configure the first Directory Server instance.

The `setup-ds-admin.pl` script has an option, `-f`, which references an `inf`. The `inf` can be used to import LDIF files through the `ConfigFile` parameter, and the LDIF files can create

databases, suffixes, and replication entries. (The `inf` file is described in more detail in the *Directory Server Installation Guide*.)

```
/usr/sbin/setup-ds-admin.pl -f /tmp/server1.inf
```

To configure the `o=NetscapeRoot` database on `server1` as a multi-master supplier replica, use the following statements in the `inf` file:

```
[slapd]
...
ConfigFile = repluser.ldif example supplier bind DN entry
ConfigFile = changelog.ldif example changelog entry
ConfigFile = replica.ldif example replica entry
ConfigFile = replagreement.ldif example replication agreement entry
...
```

2. Install and configure the second Directory Server instance. For the second server, `server2.example.com`, use the `setup-ds.pl` command, which installs a Directory Server instance without installing a local Administration Server.

```
/usr/sbin/setup-ds.pl -f /tmp/server2.inf
```

With `server2`, use the `inf` file to create and configure a `o=NetscapeRoot` database on `server2` as a multi-master supplier replica:

```
[slapd]
...
ConfigFile = netscaperootdb.ldif example suffix entry
ConfigFile = repluser.ldif example supplier bind DN entry
ConfigFile = changelog.ldif example changelog entry
ConfigFile = replica.ldif example replica entry
ConfigFile = replagreement.ldif example replication agreement entry
...
```

3. Initialize the `o=NetscapeRoot` database on `server2` from `server1`. Add the `nsds5replicarefresh` attribute to the replication agreement on `server1`.

```
ldapmodify -h supplier1.example.com -p 389 -D "cn=directory manager" -w
password

dn: cn=ExampleAgreement1,cn=replica,cn="o=NetscapeRoot",cn=mapping
tree,cn=config
changetype: modify
replace: nsds5beginreplicarefresh
nsds5beginreplicarefresh: start
```

4. Run the `register-ds-admin.pl` to create a local Administration Server on `server2` and

switch the configuration directory for `server2` to its own `o=NetscapeRoot` database from `server1`.

```
/usr/sbin/register-ds-admin.pl
```

5. Disable the PTA Plug-in on `server2` so that it does not pass bind operations for the administrative users in its `o=NetscapeRoot` to `server1`.

See [Section 2, “Enabling and Disabling Plug-ins”](#).

15. Replication with Earlier Releases

This section provides information on how to optimize replication with earlier releases of Directory Server. Directory Server 8.0 can be involved in replication with earlier releases of Directory Server, providing the following conditions are met:

- Directory Server 8.0 is a consumer.
- The legacy suppliers can be Directory Server 4.0, 4.1, and 4.1x.

The following restrictions apply:

- A legacy Directory Server and Directory Server 8.0 cannot update the same replica. However, this version of Directory Server can have different replicas, where one is supplied by a legacy Directory Server and the other is supplied by Directory Server 8.0.
- Directory Server 8.0 cannot be a supplier for other replicas.

The main advantage of using Directory Server 8.0 as a consumer of a legacy Directory Server is to ease the migration of a replicated environment. For more information on the steps to follow to migrate a replicated environment, refer to the *Directory Server Installation Guide*.

To set up legacy replication, do the following:

1. In the Directory Server Console, click the **Configuration** tab.
2. Select the **Replication** node, and click the **Legacy Consumer Settings** tab in the right pane.
3. Check the **Enable Legacy Consumer** checkbox.

This activates the fields in the **Authentication** box.

4. Specify the supplier bind DN that the legacy supplier server will use to bind to the consumer.

Optionally, specify a password at least 8 characters long.

5. Click **Save**.
6. Now configure legacy consumer settings for each replica that will receive updates from a legacy supplier.
 - a. In the navigation tree, expand the **Replication** node, and select a replica that will receive updates from the legacy supplier.
 - b. In the **Replica Settings** tab, select the **Enable Replica** and **Updatable by a 4.x Replica** checkboxes.

These options are the only ones required for replication to work. Optionally, specify a replica ID. It is not necessary to specify a supplier DN because the one specified in step 4 will be used.
 - c. Click **Save**.
7. Repeat step 6 for each read-only replica that will receive updates from a legacy supplier.
8. To complete the legacy replication setup, configure the legacy supplier to replicate to the Directory Server 8.0 instance. For instructions on configuring a replication agreement on a 4.x Directory Server, refer to the documentation for the legacy Directory Server.



NOTE

The Directory Server Console will not prevent you from configuring a database as a read-write replica and enabling legacy consumer settings. This makes migration easier because the Directory Server can be configured as it should be after the migration and legacy consumer settings only have to be active for the duration of the transition.

16. Using the Retro Changelog Plug-in

The Retro Changelog plug-in configures Directory Server to maintain a changelog that is compatible with the changelog implemented in Directory Server 4.0, 4.1, and 4.1x. Maintaining a retro changelog is essential to maintain a changelog for directory clients that depend on a Directory Server 4.x-style changelog.

To use the retro changelog plug-in, the Directory Server 8.0 instance must be configured as a single-master replica.

When the Directory Server is configured to maintain a retro changelog, this changelog is stored in a separate database under a special suffix, `cn=changelog`.

The retro changelog consists of a single level of entries. Each entry in the changelog has the object class `changeLogEntry` and can include the attributes listed in [Table 8.5, “Attributes of a Retro Changelog Entry”](#).

Attribute	Definition
changeNumber	This single-valued attribute is always present. It contains an integer which uniquely identifies each change. This number is related to the order in which the change occurred. The higher the number, the later the change.
targetDN	This attribute contains the DN of the entry that was affected by the LDAP operation. In the case of a <code>modrdn</code> operation, the <code>targetDN</code> attribute contains the DN of the entry before it was modified or moved.
changeType	Specifies the type of LDAP operation. This attribute can have a value of add, delete, modify, or <code>modrdn</code> .
changes	For add and modify operations, contains the changes made to the entry in LDIF format.
newRDN	In the case of <code>modrdn</code> operations, specifies the new RDN of the entry.
deleteOldRdn	In the case of <code>modrdn</code> operations, specifies whether the old RDN was deleted.
newSuperior	In the case of <code>modrdn</code> operations, specifies the <code>newSuperior</code> attribute of the entry.

Table 8.5. Attributes of a Retro Changelog Entry

This section contains information on the following retro changelog items:

- [Section 16.1, “Enabling the Retro Changelog Plug-in”](#)
- [Section 16.2, “Trimming the Retro Changelog”](#)
- [Section 16.3, “Searching and Modifying the Retro Changelog”](#)
- [Section 16.4, “Retro Changelog and the Access Control Policy”](#)

16.1. Enabling the Retro Changelog Plug-in

The retro changelog plug-in configuration information is stored in the `cn=Retro Changelog Plugin,cn=plugins,cn=config` entry in `dse.ldif`. To enable the retro changelog plug-in from the command-line, do the following:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
cn: Retro Changelog Plugin
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

2. Use the `ldapmodify` command to import the LDIF file into the directory.

For more information on the `ldapmodify` command, see [Section 2, “Managing Entries from the Command-Line”](#) and the *Directory Server Configuration, Command, and File Reference*.

3. Restart the server.

For information on restarting the server, see [Section 3, “Starting and Stopping Servers”](#).

The retro changelog is created in the directory tree under a special suffix, `cn=changelog`.

The procedure for enabling the retro changelog plug-in from Directory Server Console is the same as for all Directory Server plug-ins. For information, see [Section 2, “Enabling and Disabling Plug-ins”](#).

16.2. Trimming the Retro Changelog

The entries in the changelog can be automatically removed after a specified period of time. To configure the period of time after which entries are automatically deleted from the changelog, set the `nsslapd-changelogmaxage` configuration attribute in the `cn=Retro Changelog Plugin,cn=plugins,cn=config` entry.

The `nsslapd-changelogmaxage` attribute is a single-valued attribute. Its syntax is as follows:

```
nsslapd-changelogmaxage: Integer timeUnit
```

Integer is a number, and *timeUnit* can be *s* for seconds, *m* for minutes, *h* for hours, *d* for days, or *w* for weeks.



NOTE

There should not be a space between the *Integer* and *timeUnit* variables. The space in the syntax above is intended to show that the attribute value is composed of two variable parts, not just one. For example:

```
nsslapd-changelogmaxage: 2d
```

16.3. 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.

16.4. 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.

17. Monitoring Replication Status

The replication status can be viewed in the Directory Server Console or Red Hat Administration Express.

- [Section 17.1, “Monitoring Replication Status from the Directory Server Console”](#)
- [Section 17.2, “Monitoring Replication Status from Administration Express”](#)

17.1. Monitoring Replication Status from the Directory Server Console

To view a summary of replication status in the Directory Server Console, do the following:

1. Open the Directory Server Console.
2. Select the **Status** tab, and then, in the left navigation tree, select **Replication Status**.

In the right pane, a table appears that contains information about each of the replication

agreements configured for this server.

3. Click **Refresh** to update the contents of the tab.

The status information displayed is described in [Table 8.6, “Directory Server Console Replication Status”](#).

Table Header	Description
Agreement	The name of the replication agreement.
Replica suffix	The suffix that is replicated.
Supplier	The supplier server in the agreement.
Consumer	The consumer server in the agreement.
Number of changes	The number of changes sent to this replica since the server started.
Last replica update began	The time when the most recent replication update started.
Last replica update ended	The time when the most recent replication update ended.
Last update message	The status for the most recent replication updates.
Consumer initialization	The current status on consumer initialization (in progress or not).
Last consumer initialization update message	The status on the last initialization of the consumer.
Last consumer initialization began	The time when the initialization of the consumer replica started.
Last consumer initialization ended	The time when the initialization of the consumer replica ended.

Table 8.6. Directory Server Console Replication Status

17.2. Monitoring Replication Status from Administration Express

Although the replication status report in the Directory Server Console shows many details, it does not show the progress of the replication. Additionally, because one report is generated per agreement, this lists all status reports for all different agreements.

The `repl-monitor.pl` script, which is explained in detail in the *Directory Server Configuration, Command, and File Reference*, monitors replication status to a greater extent by providing these functionalities:

- Lists for each supplier replica on each Directory Server discovered, server URL or alias, replica ID, replica root, and maximum change sequence number (`maxcsn`).
- Lists corresponding to each supplier replica listed above and for each direct or indirect consumer replicas discovered, server URL or alias, replica root, replica type, connection type of the replication sessions, replication schedule, replication status, supplier `maxcsn`, and time lag between the consumer `maxcsn` and the supplier `maxcsn`.

The time lag field uses different colors to indicate the degree of time lag. The threshold for each color is configurable.

- Displays the change sequence number (CSN) in human-readable format (instead of hex strings) in the `MM/DD/YYYY HH:MI Seq# SubSeq#` format, where `Seq#` and `SubSeq#` are omitted if they are zero.
- Shows the output/result in the HTML format. The script writes the output to an HTML file, which can be configured to refresh itself automatically; the refresh interval is also configurable.

The script is integrated into the Red Hat Administration Express, so the replication status can be viewed in a web browser. The Administration Express is an HTML-based version of Red Hat Console that provides quick access to servers running Administration Server.

To view in-progress status of replication in Administration Express, do the following:

1. Prepare a configuration file following the guidelines provided in the "repl-monitor.pl (Monitor replication status)" section of the *Directory Server Configuration, Command, and File Reference*.
2. Open the Administration Server URL in a web browser.

```
http://hostname:admin_port
```

3. Click **Red Hat Administration Express**, and, when prompted, log in.
4. Select a supplier Directory Server instance, and click **Replication Status**.

This brings up a page for specifying the runtime parameters of the replication-monitoring tool.

5. In the **Configuration file** field, type the path to the configuration file created in step 1, and click **OK**.

The replication-status page appears; by default, the page gets refreshed every 300 seconds.

Each table shows the status of the changes originated from a supplier replica.

Table	Description
Table Header	The table header shows the replica ID of the

Table	Description
	supplier replica, the replica root, and the maximum Change State Number (CSN) on the supplier. The important thing is to make sure that each supplier LDAP server has its unique replica ID. Multiple replica roots on one LDAP server, however, could share the same replica ID.
Table Row	Each row represents a direct or indirect consumer of the supplier (identified in the Table Header).
Max CSN	It is the most recent CSN the consumer has replayed that was originated from the supplier (identified in the Table Header).
Time Lag	It shows the time difference between the supplier and the consumer's max CSNs for the changes originated from the supplier (identified in the Table Header). A consumer is in sync with its supplier when its time lag is 0.
Last Modify Time	It is roughly the time when the consumer's max CSN was replayed.
Supplier	This column lists all the suppliers of the consumer.
Sent/Skipped	Each supplier lists roughly how many changes originated from the supplier (identified in the Table Header) have been replayed or skipped by the consumer. The numbers are kept in suppliers' memory only. They will be cleared if the supplier is restarted.
Update Status	The number is the status code, and the string is the implication of the status code. Watch this column for possible deadlock if all the suppliers complain that they cannot acquire the busy replica. It is normal if one of the suppliers is doing an update while the others can't acquire the busy replica.

18. Solving Common Replication Conflicts

Multi-master replication uses a loose consistency replication model. This means that the same entries can be changed on different servers. When replication occurs between the two servers,

the conflicting changes need to be resolved. Mostly, resolution occurs automatically, based on the timestamp associated with the change on each server. The most recent change takes precedence.

However, there are some cases where change conflicts require manual intervention in order to reach a resolution. Entries that have a change conflict that cannot be resolved automatically by the replication process contain a conflict marker attribute *nsds5ReplConflict*. The *nsds5ReplConflict* attribute is an operational attribute which is indexed for presence and equality, so it is simple to search for entries that contain this attribute. For example:

```
ldapsearch -D adminDN -w password
-b "dc=example,dc=com" "nsds5ReplConflict=*" \* nsds5ReplConflict
```

The *nsds5ReplConflict* attribute is already indexed for presence and equality, but for performance reasons, if there are many conflicting entries every day, index the *nsds5ReplConflict* attribute in other indexes. For information on indexing, see [Chapter 10, Managing Indexes](#).

This section contains the procedures for the following conflict resolution procedures:

- [Section 18.1, “Solving Naming Conflicts”](#)
- [Section 18.2, “Solving Orphan Entry Conflicts”](#)
- [Section 18.3, “Solving Potential Interoperability Problems”](#)

18.1. Solving Naming Conflicts

When two entries are created with the same DN on different servers, the automatic conflict resolution procedure during replication renames the last entry created, including the entry's unique identifier in the DN. Every directory entry includes a unique identifier given by the operational attribute *nsuniqueid*. When a naming conflict occurs, this unique ID is appended to the non-unique DN.

For example, the entry `uid=adamss,ou=people,dc=example,dc=com` is created on server A at time t_1 and on server B at time t_2 , where t_2 is greater (or later) than t_1 . After replication, server A and server B both hold the following entries:

- `uid=adamss,ou=people,dc=example,dc=com` (created at time t_1)
- `nsuniqueid=66446001-1dd211b2+uid=adamss,dc=example,dc=com` (created at time t_2)

The second entry needs to be renamed in such a way that it has a unique DN. The renaming procedure depends on whether the naming attribute is single-valued or multi-valued.

18.1.1. Renaming an Entry with a Multi-Valued Naming Attribute

To rename an entry that has a multi-valued naming attribute, do the following:

1. Rename the entry using a new value for the naming attribute, and keep the old RDN. For example:

```
ldapmodify -D adminDN -w password
dn: nsuniqueid=66446001-1dd211b2+uid=adamss,dc=example,dc=com
changetype: modrdn
newrdn: uid=NewValue
deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -D adminDN -w password
dn: uid=NewValue,dc=example,dc=com
changetype: modify
delete: uid
uid: adamss
-
delete: nsds5ReplConflict
-
```



NOTE

The unique identifier attribute *nsuniqueid* cannot be deleted.

For more information on the `ldapmodify` command, see [Section 2, “Managing Entries from the Command-Line”](#) and the *Directory Server Configuration, Command, and File Reference*.

The Console does not support editing multi-valued RDNs. For example, if there are two servers in a multi-master mode, an entry can be created on each server with the same user ID, and then the new entries' RDN changed to the *nsuniqueid uid* value. Attempting to modify this entry from the Console returns the error *Changes cannot be saved for entries with multi-valued RDNs*.

Opening the entry in the advanced mode shows that the naming attribute has been set to *nsuniqueid uid*. However, the entry cannot be changed or corrected by changing the user ID and RDN values to something different. For example, if *jdoe* was the user ID and it should be changed to *jdoe1*, it cannot be done from the Console. Instead, use the `ldapmodify` command:

```
dn: cn=John Doe
changetype: modify
```

```
replace: uid
uid: jdoe

dn: cn=John Doe
changetype: modrdn
newrdn: uid=jdoe1
deleteoldrdn: 1
```

18.1.2. Renaming an Entry with a Single-Valued Naming Attribute

To rename an entry that has a single-valued naming attribute, do the following:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:

```
ldapmodify -D adminDN -w password
dn: nsuniqueid=66446001-1dd211b2+dc=pubs,dc=example,dc=com
changetype: modrdn
newrdn: cn=TempValue
deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -D adminDN -w password
dn: cn=TempValue,dc=example,dc=com
changetype: modify
delete: dc
dc: pubs
-
delete: nsds5ReplConflict
-
```



NOTE

The unique identifier attribute *nsuniqueid* cannot be deleted.

3. Rename the entry with the intended attribute-value pair. For example:

```
ldapmodify -D adminDN -w password
dn: cn=TempValue,dc=example,dc=com
changetype: modrdn
newrdn: dc=NewValue
deleteoldrdn: 1
```

Setting the value of the `deleteoldrdn` attribute to 1 deletes the temporary attribute-value pair `cn=TempValue`. To keep this attribute, set the value of the `deleteoldrdn` attribute to 0.

For more information on the `ldapmodify` command, see [Section 2, “Managing Entries from the Command-Line”](#) and the *Directory Server Configuration, Command, and File Reference*.

18.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a `glue` entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

Glue entries are temporary entries that include the object classes `glue` and `extensibleObject`. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the `glue` object class and the `nsds5ReplConflict` attribute.

In such cases, either modify the glue entry to remove the `glue` object class and the `nsds5ReplConflict` attribute to keep the entry as a normal entry or delete the glue entry and its child entries.

- The server creates a minimalistic entry with the `glue` and `extensibleObject` object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

18.3. Solving Potential Interoperability Problems

For reasons of interoperability with applications that rely on attribute uniqueness, such as a mail server, it may be necessary to restrict access to the entries which contain the `nsds5ReplConflict` attribute. If access is not restricted to these entries, then the applications requiring one attribute only pick up both the original entry and the conflict resolution entry containing the `nsds5ReplConflict`, and operations will fail.

To restrict access, modify the default ACI that grants anonymous read access:

```
ldapmodify -h hostname -D "cn=Directory Manager" -w password
> dn: dc=example,dc=com
> changetype: modify
> delete: aci
> aci: (target = "ldap:///dc=example,dc=com")(targetattr
    != "userPassword")(version 3.0;acl "Anonymous read-search
```

```

        access";allow (read, search, compare)(userdn = "ldap:///anyone");)
> -
> add: aci
> aci: (target="ldap:///dc=example,dc=com")(targetattr!="userPassword")
      (targetfilter="(!(nsds5ReplConflict=*))")(version 3.0;acl
        "Anonymous read-search access";allow (read, search, compare)
        (userdn="ldap:///anyone");)
> -

```

The new ACI filters out all entries that contain the `nsds5ReplConflict` attribute from search results.

For more information on the `ldapmodify` command, see [Section 2, “Managing Entries from the Command-Line”](#) and the *Directory Server Configuration, Command, and File Reference*.

19. 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 19, “Troubleshooting Replication-Related Problems”](#).

To change the error log level to 8192, run the following `ldapmodify` command:

```

dn: cn=config
changetype: modify
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 8192

```

Because log level is additive, running the above command will result in excessive messages in the error log. So, use it judiciously.

To turn off replication debugging log, set the same attribute to 0.

The `template-cl-dump.pl` script, which is explained in detail in the *Directory Server Configuration, Command, and File Reference* can also help troubleshoot replication-related problems. Depending on the usage options, the script can selectively dump a particular replica:

- Dump the contents of a `replication-change-log` file and in-memory variables `purge RUV` and `maxRUV`.
- Grep and interpret change sequence numbers (CSNs) in the changelog.
- Get the base-64 encoded changelog from the Directory Server, and then decode the changelog.

Many common replication problems are described in [Table 8.7, “Replication Errors”](#).

Error/Symptom	Reason	Impact	Remedy
agmt=%s (%s:%d) Replica has a different generation ID than the local data	The consumer specified at the beginning of this message has not been (successfully) initialized yet, or it was initialized from a different root supplier.	The local supplier will not replicate any data to the consumer.	Ignore this message if it occurs before the consumer is initialized. Otherwise, reinitialize the consumer if the message is persistent. In a multi-master environment, all the servers should be initialized only once from a root supplier, directly or indirectly. For example, M1 initializes M2 and M4, M2 then initializes M3, and so on. The important thing to note is that M2 must not start initializing M3 until M2's own initialization is done (check the total update status from the M1's Console or M1 or M2's error log). Also, M2 should not initialize M1 back.
Warning: data for replica's was reloaded, and it no 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.	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	The server will not be able to send the changes to a consumer if the consumer's <i>maxcsn</i> no longer exists in the server's changelog.	Check the disk space and the possible core file (under the server's logs directory). If this is a single-master replication, reinitialize the consumers. Otherwise, if the server later complains that it can't locate some CSN for a consumer, see if the consumer can get the CSN from other

Error/Symptom	Reason	Impact	Remedy
	down.		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.	Most likely the changelog was recreated because of the disk is full or the server ungracefully shutdown.	The local server will not be able to send any more change to that consumer until the consumer is reinitialized or gets the CSN from other suppliers.	If this is a single-master replication, reinitialize the consumers. Otherwise, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.
Too much time skew	The system clocks on the host machines are extremely out of sync.	The system clock is used to generate a part of the CSN. In order to reflect the change sequence among multiple suppliers, suppliers would forward-adjust their local clocks based on the remote clocks of the other suppliers. Because the adjustment is limited to a certain amount, any difference that exceeds the permitted limit will cause the replication session to be aborted.	Synchronize the system clocks on the Directory Server host machines. If applicable, run the network time protocol (ntp) daemon on those hosts.
agmt=%s(%s:%d): Warning: Unable to send endReplication extended operation (%s)	The consumer is not responding.	If the consumer recovers without being restarted, there is a chance that the replica on the consumer will be locked forever if it did not receive the release lock message from the supplier.	Watch if the consumer can receive any new change from any of its suppliers, or start the replication monitor, and see if all the suppliers of this consumer warn that the replica is busy. If the replica appears to

Error/Symptom	Reason	Impact	Remedy
			be locked forever and no supplier can get in, restart the consumer.
Changelog is getting too big.	Either changelog purge is turned off, which is the default setting, or changelog purge is turned on, but some consumers are way behind the supplier.		<p>By default changelog purge is turned off. To turn it on from the command-line, run <code>ldapmodify</code> as follows:</p> <pre>dn: cn=changelog5,cn=config changetype: modify add: nsslapd-changelogmaxage nsslapd-changelogmaxage: 1d</pre> <p>where <code>1d</code> means 1 day. Other valid time units are <code>s</code> for seconds, <code>m</code> for minutes, <code>h</code> for hours, and <code>w</code> for weeks. A value of <code>0</code> turns off the purge.</p> <p>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 <code>nsslapd-changelogmaxage</code> and if it has been replayed to all the direct consumers of this supplier (supplier or hub).</p> <p>If it appears that the changelog is not purged when the purge threshold is</p>

Error/Symptom	Reason	Impact	Remedy
			reached, check the maximum time lag from the replication monitor among all the consumers. Irrespective of what the purge threshold is, no change will be purged before it is replayed by all the consumers.
The Replication Monitor is not responding. (For information on Replication Monitor, see Section 17, "Monitoring Replication Status" .)	The SSL port is specified in some replication agreement, but the certificate database is not specified or not accessible by the Replication Monitor. If there is no SSL port problem, one of the servers in the replication topology might hang.		Map the SSL port to a non-SSL port in the configuration file of the Replication Monitor. For example, if 636 is the SSL port and 389 is the non-SSL port, add the following line in the [connection] section: *:636=389:*:password
In the Replication Monitor, some consumers show just the header of the table. (For information on Replication Monitor, see Section 17, "Monitoring Replication Status" .)	No change has originated from the corresponding suppliers. In this case, the MaxCSN: in the header part should be "None".		There is nothing wrong if there is no change originated from a supplier.

Table 8.7. Replication Errors

Extending 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.

1. Overview of Extending Schema

There are two steps, in the proper order, required to extend the directory schema:

1. Create new attributes, as in [Section 2.2, “Creating Attributes”](#).
2. Create an object class to contain the new attributes, and add the attributes to the object class, as in [Section 3.2, “Creating Object Classes”](#).

When new attributes are added to the schema, a new object class must be created to contain them. Although it may seem convenient to add any new attributes to an existing object class that already contains most of the attributes you require, doing so compromises interoperability with LDAP clients. The interoperability of Directory Server with existing LDAP clients relies on the standard LDAP schema. Changing the standard schema can also create difficulties when upgrading the Directory Server. For these reasons, standard schema elements, both attributes and object classes, cannot be edited or deleted.

2. Managing Attributes

The Directory Server Console shows all attributes in the schema, and you can create, edit, and delete attribute extensions to the schema. The following sections describe how to manage attributes:

- [Section 2.1, “Viewing Attributes”](#)
- [Section 2.2, “Creating Attributes”](#)
- [Section 2.3, “Editing Attributes”](#)
- [Section 2.4, “Deleting Attributes”](#)

For information on managing object classes, see [Section 3, “Managing Object Classes”](#).


2.1. Viewing Attributes

To view information about all attributes that currently exist in the directory schema, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder, and then select the **Attributes** tab in the right pane.

This tab contains information about all the standard (read-only) and user-defined attributes in the schema.

The fields and lists in the **Attributes** tab are described in [Table 9.1, “Attributes Tab Reference”](#).

Field	Description
Name	The name of the attribute.
OID	<p>The object identifier of the attribute. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If an OID is not specified, the Directory Server automatically uses <i>attribute_name-oid</i>. For example, if the attribute <i>birthdate</i> is created without supplying an OID, the Directory Server automatically uses <i>birthdate-oid</i> as the OID.</p> <div>CAUTION<p>Using an alphanumeric OID such as <i>birthdate-oid</i> is deprecated and strongly discouraged because it will lead to interoperability problems. Red Hat strongly encourages that you request your own OID prefix from IANA (Internet Assigned Number Authority). For more information about OIDs or to request a prefix, email IANA at mailto:iana@iana.org, or visit the IANA website at http://www.iana.org/.</p></div>
Syntax	Sets the syntax for the attribute values. The attribute syntax can be, for example, any of the following:

Field	Description
	<p>Case Ignore String — Values for this attribute are not case-sensitive.</p> <p>Case Exact String — Values for this attribute are case-sensitive.</p> <p>Distinguished Name — Values for this attribute are DNs.</p> <p>Binary — Values for this attribute are binary.</p> <p>Telephone Number — Values for this attribute are in telephone number format.</p> <p>Integer — Values for this attribute are numbers.</p> <p>There are many more syntaxes available for attributes.</p>
Multi	If the attribute is multi-valued, an X appears in this column; otherwise, this field is blank. If this is not checked, the Directory Server only allows a single value for this attribute.

Table 9.1. Attributes Tab Reference

2.2. Creating Attributes

The Directory Server Console can create new attributes.



NOTE

After adding new attributes to the schema, create a new object class to contain them, as described in [Section 3.2, “Creating Object Classes”](#).

To create a new attribute, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder, and then select the **Attributes** tab in the right pane.
3. Click **Create**.

The **Create Attribute** dialog box opens.

4. Enter a unique name for the attribute in the **Attribute Name** text box.

5. Enter an object identifier for the attribute in the **Attribute OID (Optional)** text box.

OIDs are described in [Table 9.1, “Attributes Tab Reference”](#).
6. Select a syntax that describes the data to be held by the attribute from the Syntax drop-down menu.

Available syntaxes are described in [Table 9.1, “Attributes Tab Reference”](#).
7. To make the attribute multi-valued, select the **Multi-Valued** checkbox.

Multi-valued means that the Directory Server allows more than one instance of the attribute per entry.
8. Click **OK**.

2.3. Editing Attributes

Only user-created attributes can be edited. You cannot edit standard attributes. To edit an attribute, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder, and then select the **Attributes** tab in the right pane.
3. Select the attribute to edit in the **User Defined Attributes** table, and click **Edit**.

The **Edit Attribute** dialog box opens.

- a. To change the attribute's name, enter a new one in the **Attribute Name** text box.
 - b. To change the attribute's object identifier, enter a new one in the **Attribute OID (Optional)** text box. OIDs are described in [Table 9.1, “Attributes Tab Reference”](#).
 - c. To change the syntax that describes the data to be held by the attribute, choose a new one from the **Syntax** drop-down menu. Available syntaxes are described in [Table 9.1, “Attributes Tab Reference”](#).
 - d. To make the attribute multi-valued, select the **Multi-Valued** checkbox. Multi-valued means that the Directory Server allows more than one instance of the attribute per entry.
4. Click **OK**.

2.4. Deleting Attributes

Only user-defined attributes can be deleted. You cannot delete standard attributes. To delete an

attribute, do the following:

1. Display the **Attributes** tab.

This procedure is explained in [Section 2.4, “Deleting Attributes”](#).

2. In the **User Defined Attributes** table, select the attribute, and click **Delete**.
3. If prompted, confirm the delete.



WARNING

The server immediately deletes the attribute. There is no undo.

3. Managing Object Classes

The Directory Server Console can manage and show the directory schema's object classes. You can view all of the current object classes and create, edit, and delete object class extensions to the schema. The following sections describe how to manage object classes:

- [Section 3.1, “Viewing Object Classes”](#)
- [Section 3.2, “Creating Object Classes”](#)
- [Section 3.3, “Editing Object Classes”](#)
- [Section 3.4, “Deleting Object Classes”](#)

For information on managing attributes, see [Section 2, “Managing Attributes”](#).


3.1. Viewing Object Classes

To view information about all object classes that currently exist in the directory schema, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.
3. In the **Object Classes** list, select the object class to view.

This tab displays information about the standard or user-defined object class selected.

The fields and lists in the **Object Classes** tab are described in [Table 9.2, “Object Classes Tab Reference”](#).

Field	Description
Parent	<p>The parent identifies the object class from which this object class inherits its attributes and structure. For example, the parent object for the <code>inetOrgPerson</code> object class is the <code>organizationalPerson</code> object. That means that an entry with the object class <code>inetOrgPerson</code> must also include the object class <code>organizationalPerson</code>.</p> <p>Typically, to add new attributes for user entries, the parent is the <code>inetOrgPerson</code> object class.</p> <p>To add new attributes for corporate entries, the parent is usually <code>organization</code> or <code>organizationalUnit</code>.</p> <p>To add new attributes for group entries, the parent is usually <code>groupOfNames</code> or <code>groupOfUniqueNames</code>.</p>
OID	<p>The object identifier of the attribute. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If an OID is not specified, the Directory Server automatically uses <code>object-class_name-oid</code>. For example, if the object <code>division</code> is created without supplying an OID, the Directory Server automatically uses <code>division-oid</code> as the OID.</p> <div>CAUTION<p>Using an alphanumeric OID such as <code>division-oid</code> is deprecated and strongly discouraged because it will lead to interoperability problems. Red Hat strongly encourages that you request your own OID prefix from IANA (Internet Assigned Number Authority). For more information about OIDs or to</p></div>

Field	Description
	<div> request a prefix, email IANA at mailto:iana@iana.org, or visit the IANA website at http://www.iana.org/. </div>
Object Classes	Lists all of the standard and user-defined object classes in the Directory Server schema.
Required Attributes	Contains a list of attributes that must be present in entries that use this object class, including inherited attributes.
Allowed Attributes	Contains a list of attributes that may be present in entries that use this object class, including inherited attributes.

Table 9.2. Object Classes Tab Reference

3.2. Creating Object Classes

A new object class must be created with a unique name, a parent object, and required and optional attributes. To create an object class, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.
3. In the **Object Classes** list, select the object class.
4. Click **Create** in the **Object Classes** tab.

The **Create Object Class** dialog box opens.

5. Enter a unique name for the object class in the **Name** text box.
6. Enter an object identifier for the new object class in the **OID (Optional)** text box.
OIDs are described in [Table 9.2, “Object Classes Tab Reference”](#).
7. Select a parent object for the object class from the **Parent** drop-down menu.

Any existing object class can be the parent of the new object class. See [Table 9.2, “Object Classes Tab Reference”](#) for more information on parent object classes.

8. To add an attribute that *must* be present in entries that use the new object class, highlight the attribute in the **Available Attributes** list, and then click the **Add** button to the left of the **Required Attributes** box.

Both standard attributes and user-defined are allowed. For information on creating custom attributes, see [Section 2.2, “Creating Attributes”](#).

9. To add an attribute that may optionally be present in entries that use the new object class, highlight the attribute in the **Available Attributes** list, and then click the **Add** button to the left of the **Allowed Attributes** box.
10. To remove an attribute belonging to the object class, highlight the attribute in the **Required Attributes** list or the **Allowed Attributes** list, and then click the **Remove** button.



NOTE

Attributes that are inherited from the parent object classes cannot be removed, regardless of whether they are allowed or required.

11. Click **OK** to save the new object class.

3.3. Editing Object Classes

Only user-defined object classes can be edited. You cannot edit a standard object class. To edit a user-defined object class in the Directory Server Console, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.
3. Select the object class to edit from the **Object Classes** list, and click **Edit**.

The **Edit Object Class** dialog box opens.

- a. To change the name of the object class, enter the new name in the **Name** text box.
- b. To change the object identifier for the object class, enter the new OID in the **OID (Optional)** text box. OIDs are described in [Table 9.2, “Object Classes Tab Reference”](#).

- c. To change the parent object for the object class, select the new parent from the **Parent** pull-down menu.
- d. To add an attribute that must be present in entries that use the new object class, highlight the attribute in the **Available Attributes** list, and then click the **Add** button to the left of the **Required Attributes** box.

Both standard attributes and user-defined attributes are allowed for the object class. Creating custom attributes is described in [Section 2.2, "Creating Attributes"](#).
- e. To add an attribute that may be present in entries that use the new object class, highlight the attribute in the **Available Attributes** list, and then click the **Add** button to the left of the **Allowed Attributes** box.
- f. To remove an attribute from the object class, highlight the attribute in the **Required Attributes** list or the **Allowed Attributes** list, and then click the **Remove** button.

**NOTE**

Inherited attributes cannot be removed, regardless of whether they are required or allowed.

- 4. Click **OK**.

3.4. Deleting Object Classes

Only user-defined object classes can be deleted. You cannot delete standard object classes. To delete an object class, do the following:

- 1. In the Directory Server Console, select the **Configuration** tab.
- 2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.
- 3. Select the object class to remove, and click **Delete**.
- 4. If prompted, confirm the delete.

**WARNING**

The server immediately deletes the object class. There is no undo.

4. Turning Schema Checking On and Off

When schema checking is on, the Directory Server ensures three things:

- The object classes and attributes using are defined in the directory schema.
- The attributes required for an object class are contained in the entry.
- Only attributes allowed by the object class are contained in the entry.

Schema checking is turned on by default in the Directory Server, and the Directory Server should always run with schema checking turned on. The only situation where 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 search for these entries.

To turn schema checking on and off, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. Highlight the server icon at the top of the navigation tree, then select the **Settings** tab in the right pane.
3. To enable schema checking, check the **Enable Schema Checking** checkbox; clear it to turn off schema checking.
4. Click **Save**.

To turn schema checking on and off using LDAP commands, edit the value of the `nsslapd-schemacheck` attribute. For example:

```
ldapmodify -h myserver -p 389 -D "cn=directory manager" -w secretpwd  
  
dn: cn=config  
changetype: modify  
replace: nsslapd-schemacheck: on  
nsslapd-schemacheck: off
```

For information, see the *Directory Server Configuration, Command, and File Reference*.

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.

1. About Indexes

This section provides an overview of indexing in Directory Server. It contains the following topics:

- [Section 1.1, “About Index Types”](#)
- [Section 1.2, “About Default, System, and Standard Indexes”](#)
- [Section 1.3, “Overview of the Searching Algorithm”](#)
- [Section 1.5, “Balancing the Benefits of Indexing”](#)

1.1. About Index Types

Indexes are stored in files in the directory's databases. The names of the files are based on the indexed attribute, not the type of index contained in the file. Each index file may contain multiple types of indexes if multiple indexes are maintained for the specific attribute. For example, all indexes maintained for the common name attribute are contained in the `cn.db4` file.

Directory Server supports the following types of index:

- *Presence index (pres)* contains a list of the entries that contain a particular attribute, which is very useful for searched. For example, it makes it easy to examine any entries that contain access control information. Generating an `aci.db4` file that includes a presence index efficiently performs the search for `ACI=*` to generate the access control list for the server.

The presence index is not used for base object searches.
- *Equality index (eq)* improves searches for entries containing a specific attribute value. For example, an equality index on the `cn` attribute allows a user to perform the search for `cn=Babs Jensen` far more efficiently.
- *Approximate index (approx)* is used for efficient approximate or *sounds-like* searches. For example, an entry may include the attribute value `cn=Robert E Lee`. An approximate search would return this value for searches against `cn~=Robert Lee`, `cn~=Robert`, or `cn~=Lee`. Similarly, a search against `l~=San Fransisco` (note the misspelling) would return entries

including `l=San Francisco`.

- *Substring index (sub)* is a costly index to maintain, but it allows efficient searching against substrings within entries. Substring indexes are limited to a minimum of three characters for each entry.

For example, searches of the form `cn=*derson` , match the common names containing strings such as `Bill Anderson`, `Jill Henderson`, or `Steve Sanderson`. Similarly, the search for `telephonenumber= *555*` returns all the entries in the directory with telephone numbers that contain 555.

- *International index* speeds up searches for information in international directories. The process for creating an international index is similar to the process for creating regular indexes, except that it applies a *matching rule* by associating an *object identifier* (OID) with the attributes to be indexed.

The supported locales and their associated OIDs are listed in [Appendix D, Internationalization](#). If to configure the Directory Server to accept additional matching rules, contact Red Hat Professional Services.

- *Browsing index, or virtual list view (VLV) index*, speeds up the display of entries in the Directory Server Console. This index is particularly useful if a branch of your directory contains hundreds of entries; for example, the `ou=people` branch. You can create a browsing index on any branch point in the directory tree to improve display performance through the Directory Server Console or by using the `vlvindex` command-line tool, which is explained in the *Directory Server Configuration, Command, and File Reference*.

1.2. About Default, System, and Standard Indexes

When you install Directory Server, a set of default and system indexes is created per database instance. To maintain these indexes, the directory uses standard indexes.

1.2.1. Overview of Default Indexes

The default indexes can be modified depending on the directory indexing needs. Always ensure that no server plug-ins or other servers depend on a default index before removing it.

Table 10.1, “Default Indexes” lists the default indexes installed with the directory.

Attribute	Eq	Pres	Sub	Purpose
cn	●	●	●	Improves the performance of the most common types of user directory searches.

Attribute	Eq	Pres	Sub	Purpose
givenName	●	●	●	Improves the performance of the most common types of user directory searches.
mail	●	●	●	Improves the performance of the most common types of user directory searches.
mailHost	●			Used by a messaging server.
member	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 5, “Maintaining Referential Integrity” for more information.
owner	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 5, “Maintaining Referential Integrity” for more information.
see Also	●			Improves Directory Server

Attribute	Eq	Pres	Sub	Purpose
				performance. This index is also used by the Referential Integrity Plug-in. See Section 5, “Maintaining Referential Integrity” for more information.
sn	●	●	●	Improves the performance of the most common types of user directory searches.
telephoneNumber	●	●	●	Improves the performance of the most common types of user directory searches.
uid	●			Improves Directory Server performance.
unique member	●			Improves Directory Server performance. This index is also used by the Referential Integrity Plug-in. See Section 5, “Maintaining Referential Integrity” for more information.

Table 10.1. Default Indexes

1.2.2. Overview of System Indexes

System indexes cannot be deleted or modified. They are required by the directory to function properly. [Table 10.2, “System Indexes”](#) lists the system indexes included with the directory.

Attribute	Eq	Pres	Purpose
aci		●	Allows the Directory Server to quickly obtain the access control information maintained in the database.
objectClass	●		Used to help accelerate subtree searches in the directory.
entryDN	●		Speeds up entry retrieval based on DN searches.
parentID	●		Enhances directory performance during one-level searches.
numSubordinates		●	Used by the Directory Server Console to enhance display performance on the Directory tab.
nsUniqueID	●		Used to search for specific entries.

Table 10.2. System Indexes

1.2.3. Overview of Standard Indexes

Because of the need to maintain default indexes and other internal indexing mechanisms, the Directory Server also maintains certain standard index files. The standard index, `id2entry.db4`, exists by default in Directory Server; you do not need to generate it.

The `id2entry.db4` contains the actual directory database entries. All other database files can be recreated from this one.

1.3. Overview of the Searching Algorithm

Indexes are used to speed up searches. To understand how the directory uses indexes, it helps

to understand the searching algorithm. Each index contains a list of attributes (such as the `cn`, common name, attribute) and a pointer to the entries corresponding to each value. Directory Server processes a search request as follows:

1. An LDAP client application, such as the Directory Server Gateway, sends a search request to the directory.
2. The directory examines the incoming request to make sure that the specified base DN matches a suffix contained by one or more of its databases or database links.
 - If they do match, the directory processes the request.
 - If they do not match, the directory returns an error to the client indicating that the suffix does not match. If a referral has been specified in the `nsslapd-referral` attribute under `cn=config`, the directory also returns the LDAP URL where the client can attempt to pursue the request.
 - If the search request for each database attribute can be satisfied by a single index, then the server reads that index to generate a list of potential matches.
 - If there is no index for the attribute, the directory generates a candidate list that includes all entries in the database, which makes the search considerably slower.
 - If a search request contains multiple attributes, the directory consults multiple indexes and then combines the resulting lists of candidate entries.
 - If there is an index for the attribute, the directory takes the candidate matches from the index files in the form of a series of entry ID numbers.
3. The directory uses the returned entry ID numbers to read the corresponding entries from the `id2entry.db4` file. The Directory Server then examines each of the candidate entries to see if any match the search criteria. The directory returns matching entries to the client as each is found.

The directory continues until either it has examined all candidate entries or it reaches the limit set in one of the following attributes:

- `nsslapd-sizelimit` which specifies the maximum number of entries to return from a search operation. If this limit is reached, the directory returns any entries it has located that match the search request, as well as an exceeded size limit error.
- `nsslapd-timelimit` which specifies the maximum number of seconds allocated for a search request. If this limit is reached, the directory returns any entries it has located that match the search request, as well as an exceeded time limit error.
- `nsslapd-lookthroughlimit`, which specifies the maximum number of entries that the

directory will check when examining candidate entries in response to a search request.

- `nsslapd-idlistscanlimit` which specifies the maximum number of entries in an ID list before the list is considered to equal the entire database.

See *Directory Server Configuration, Command, and File Reference* for further information about these attributes.

1.4. Approximate Searches

In addition, the directory uses a variation of the metaphone phonetic algorithm to perform searches on an approximate index. Each value is treated as a sequence of words, and a phonetic code is generated for each word.



NOTE

The metaphone phonetic algorithm in Directory Server supports only US-ASCII letters. Therefore, use approximate indexing only with English values.

Values entered on an approximate search are similarly translated into a sequence of phonetic codes. An entry is considered to match a query if both of the following are true:

- All of the query string codes match the codes generated in the entry string.
- All of the query string codes are in the same order as the entry string codes.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
Alice B Sarette (ALS B SRT)	Alice Sarette (ALS SRT)	Matches. Codes are specified in the correct order.
	Alice Sarrette (ALS SRT)	Matches. Codes are specified in the correct order, despite the misspelling of Sarette.
	Surette (SRT)	Matches. The generated code exists in the original name, despite the misspelling of Sarette.
	Bertha Sarette (BR0 SRT)	No match. The code BR0 does not exist in the original name.
	Sarette, Alice (SRT ALS)	No match. The codes are not

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
		specified in the correct order.

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 the Directory Server generates the new index entries.
4. Once the server completes the indexing, the actual attribute values are created according to the client request.

For example, the Directory Server adds the entry:

```
dn: cn=John Doe, ou=People,dc=example,dc=com
objectclass: top
objectClass: person
objectClass: orgperson
objectClass: inetorgperson
cn: John Doe
cn: John
sn: Doe
```

```
ou: Manufacturing
ou: people
telephonenumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.
```

The Directory Server is maintaining 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.

2. Creating Indexes

This section describes how to create presence, equality, approximate, substring, and international indexes for specific attributes using the Directory Server Console and the command-line.



NOTE

Because Directory Server 8.0 can operate in either a single or multi-database environment, remember to create new indexes in every database instance since newly-created indexes are not automatically created in the other databases. However, the same is not true for default indexes because they are automatically present and maintained in subsequent database instances but not added to existing ones. In other words, the directory uses the most recently-created set of default indexes in subsequent databases. This means that if you add a default index to your second database instance, it will not be maintained in your first database instance but will be maintained in any subsequent instances.



NOTE

The procedure for creating browsing indexes is different than for creating other index types; that procedure is covered in [Section 2.3, “Creating Browsing Indexes from the Server Console”](#).

- [Section 2.1, “Creating Indexes from the Server Console”](#)
- [Section 2.2, “Creating Indexes from the Command-Line”](#)
- [Section 2.3, “Creating Browsing Indexes from the Server Console”](#)
- [Section 2.4, “Creating Browsing Indexes from the Command-Line”](#)

2.1. Creating Indexes from the Server Console

This section describes how to create presence, equality, approximate, substring, and international indexes for specific attributes using the Directory Server Console.

To create any of these indexes, do the following:

1. Select the **Configuration** tab.
2. Expand the **Data** node, expand the suffix of the database to index, and select the database.
3. Select the **Indexes** tab in the right pane.

**NOTE**

Do not click the **Database Settings** node because this opens the **Default Index Settings** window, not the window for configuring indexes per database.

4. If the attribute to be indexed is listed in the **Additional Indexes** table, go to step 6. Otherwise, click **Add Attribute** to open a dialog box with a list of all of the available attributes in the server schema.

5. Select the attribute you want to index, and click **OK**.

The server adds the attribute to the **Additional Indexes** table.

6. Select the checkbox for each type of index to maintain for each attribute.

7. To create an index for a language other than English, enter the OID of the *collation order* to use in the **Matching Rules** field.

To index the attribute using multiple languages, list multiple OIDs separated by commas, but no whitespace. For a list of languages, their associated OIDs, and further information regarding collation orders, see [Appendix D, Internationalization](#).

8. Click **Save**.

9. The **Indexes** dialog box appears, displaying the status of the index creation and informing you when the indexes have been created. Click the **Status Logs** box to view the status of the indexes created. Once the indexing is complete, click **Close**.

The new index is immediately active for any new data that you add and any existing data in your directory. You do not have to restart your server.

2.2. Creating Indexes from the Command-Line

Creating presence, equality, approximate, substring, and international indexes for specific attributes from the command-line involves two steps:

1. Using the `ldapmodify` command-line utility to add a new index entry or edit an existing index entry. See [Section 2.2.1, “Adding an Index Entry”](#).
2. Running the `db2index.pl` Perl script to generate the new set of indexes to be maintained by the server. See [Section 2.2.2, “Running the db2index.pl Script”](#).



NOTE

You cannot create new system indexes because system indexes are hard-coded in Directory Server.

2.2.1. Adding an Index Entry

Use `ldapmodify`¹ to add the new index attributes to your directory.

- To create a new index that will become one of the default indexes, add the new index attributes to the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.
- To create a new index for a particular database, add it to the `cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=database_name` corresponds to the name of the database.



NOTE

Avoid creating entries under `cn=config` in the `dse.ldif` file. The `cn=config` entry in the simple, flat `dse.ldif` configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under `cn=config`, performance will probably suffer. Although we recommend you do not store simple user entries under `cn=config` for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under `cn=config` since this centralizes configuration information.

For information on the LDIF update statements required to add entries, see [Section 4, “LDIF Update Statements”](#).

For example, to create presence, equality, and substring indexes for the `sn` (surname) attribute in the `Example1` database, do the following:

¹ The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

1. Open the Directory Server LDAP tool directory.¹

```
cd /usr/lib/mozldap
```

2. Run `ldapmodify`.

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w password
```

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

3. Add the LDIF entry for the new indexes:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

The `cn` attribute contains the name of the attribute to index, in this example the `sn` attribute. The entry is a member of the `nsIndex` object class. The `nsSystemIndex` attribute is `false`, indicating that the index is not essential to Directory Server operations. The multi-valued `nsIndexType` attribute specifies the presence (`pres`), equality (`eq`) and substring (`sub`) indexes. Each keyword has to be entered on a separate line. The `nsMatchingRule` attribute specifies the OID of the Bulgarian collation order.

Specifying an index entry with no value in the `nsIndexType` attribute results in all indexes (except international) being maintained for the specified attribute. For example, the following entry creates all index types for the `sn` index.


```
dn: cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:
```

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 collation orders and their OIDs, see [Appendix D, Internationalization](#), and for the index configuration attributes or the `ldapmodify` command-line utility, see the *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. See [Table 10.7, “Attribute Name Quick Reference Table”](#) for a list of all primary and alias attribute names.

2.2.2. Running the `db2index.pl` Script

After creating an indexing entry or added additional index types to an existing indexing entry, run the `db2index.pl` script to generate the new set of indexes to be maintained by the Directory Server. After the script is run, the new set of indexes is active for any new data added to the directory and any existing data in the directory.

To run the `db2index.pl` Perl script, do the following:

- 1. Open the Directory Server instance directory. ²

```
cd /usr/lib/dirsrv/slapd-instance_name
```

- 2. Run the `db2index.pl` Perl script.

```
db2index.pl-D "cn=Directory Manager" -w password -n ExampleServer -t sn
```

For more information about using this Perl script, see the *Directory Server Configuration, Command, and File Reference*.

[Table 10.6, “db2index Options”](#) describes the `db2index.pl` options:

Option	Description
² <code>-D</code>	This is the location for Red Hat Enterprise Linux. File locations for other platforms are listed in Section 4, “Directory Server File Locations” . Specifies the DN of the administrative user.

Option	Description
-w	Specifies the password of the administrative user.
-n	Specifies the name of the database being indexed.
-t	Specifies the name of the attribute contained by the index.

Table 10.3. db2index.pl Options

2.3. Creating Browsing Indexes from the Server Console

A virtual list view (VLV) index is a way of creating a truncated list for faster searching while enhancing server performance. The VLV index itself can be resource-intensive to maintain, but it can be beneficial in large directories (over 1000 entries).

A browsing index is a type of VLV index that organizes the entries listed into alphabetical order, making it easier to find entries. To create a browsing index using the Directory Server Console, do the following:

1. Select the **Directory** tab.
2. In the left navigation tree, select the entry, such as `People`, for which to create the index.
3. From the **Object** menu, select **Create Browsing Index**.

The **Create Browsing Index** dialog box appears displaying the status of the index creation. Click the **Status Logs** box to view the status of the indexes created.

4. Click **Close**.

The new index is immediately active for any new data that is added to the directory. You do not have to restart your server.

For more information on how to change the VLV search information or the access control rules that are set by default for VLV searches, see [Section 2.4.1, “Adding a Browsing Index Entry”](#) and [Section 2.4.3, “Setting Access Control for VLV Information”](#).

2.4. 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 2.4.1, “Adding a Browsing Index Entry”](#).

2. Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server. See [Section 2.4.2, “Running the vlvindex Script”](#).
3. Ensuring that access control on VLV index information is set appropriately. See [Section 2.4.3, “Setting Access Control for VLV Information”](#).

2.4.1. 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 [Appendix B, Finding Directory Entries](#).

- The LDBM database to which the entry that forms the base of the search belongs. You can only create browsing indexes in LDBM databases.

There is more information on `ldapsearch` options in the *Directory Server Configuration, Command, and File Reference*.

For example, create a browsing index to accelerate an `ldapsearch` on the entry `ou=People,dc=example,dc=com` held in the `Example1` database with the following attributes:

- The search base is `ou=People,dc=example,dc=com`
- The search filter is `(|(objectclass=*)(objectclass=ldapsubentry))`
- The scope is `one`
- The sorting order for the returned attributes is `cn, givenName, o, ou, and sn`

1. Run `ldapmodify`.¹

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w password
```

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

file.

2. Add an entry which specifies the base, scope, and filter of the browsing index:

```
dn: cn=MCC ou=People dc=example dc=com, cn=userRoot, cn=ldbm database,
cn=plugins, cn=config
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People, dc=example,dc=com
vlvScope: 1
vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))
```

- The *cn* contains the browsing index identifier, which specifies the entry on which to create the browsing index; in this example, the *ou=People,dc=example,dc=com* entry. Red Hat recommends using the *dn* of the entry for the browsing index identifier, which is the approach adopted by the Directory Server Console, to prevent identical browsing indexes from being created. The entry is a member of the *vlvSearch* object class.
- The *vlvbase* attribute value specifies the entry on which you want to create the browsing index; in this example, the *ou=People,dc=example,dc=com* entry (the browsing index identifier).
- The *vlvscope* attribute is 1, indicating that the scope for the search you want to accelerate is 1. A search scope of 1 means that only the immediate children of the entry specified in the *cn* attribute, and not the entry itself, will be searched.
- The *vlvfilter* specifies the filter to be used for the search; in this example, *(|(objectclass=*)(objectclass=ldapsubentry))*.

3. Add the second entry, to specify the sorting order for the returned attributes:

```
dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
dc=example dc=com, cn=userRoot, cn=ldbm database, cn=plugins,
cn= config
objectClass: top
objectClass: vlvIndex
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenName o ou sn
```

- The *cn* contains the browsing index sort identifier. The above *cn* is the type created by the Console by default, which has the sorting order as being set *by* the browsing index base. The entry is a member of the *vlvIndex* object class.
- The *vlvsort* attribute value specifies the order in which you want your attributes to be sorted; in this example, *cn, givenName, o, ou, and then sn*.



NOTE

This first browsing index entry must be added to the `cn=database_name,cn=ldbm database,cn=plugins,cn=config` directory tree node, and the second entry must be a child of the first entry.

2.4.2. Running the `vlvindex` Script

After creating the two browsing indexing entries or added additional attribute types to an existing indexing browsing entries, run the `vlvindex` script to generate the new set of browsing indexes to be maintained by the Directory Server. After running the script, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

To run the `vlvindex` script, do the following:

1. Open the Directory Server instance directory.²

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Stop the server.³

```
service dirsrv stop instance
```

3. Run the `vlvindex` script.

```
vlvindex -n Example1 -T "by MCC ou=people dc=example dc=com"
```

For more information about using this script, see the *Directory Server Configuration, Command, and File Reference*.

4. Restart the server.

```
service dirsrv start instance
```

Table 10.4, “`vlvindex` Options” describes the `vlvindex` options used in the examples:

Option	Description
<code>-n</code>	Name of the database containing the entries to index.

³ The command to stop the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, “Starting and Stopping Servers”](#).

Option	Description
-T	Browsing index identifier to use to create browsing indexes.

Table 10.4. vlvindex Options

2.4.3. Setting Access Control for VLV Information

The default access control for the VLV index information is to allow anyone who has authenticated. If a site requires anonymous users to use the VLV index information, modify the access control set for `cn: VLV Request Control` in the Directory Server's configuration.

1. Open the Directory Server configuration directory:²

```
cd /etc/dirsrv/slapd-instance_name
```

2. In a text editor, open the `dse.ldif` file.
3. Locate the `oid=2.16.840.1.113730.3.4.9` entry.

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectClass: top
objectClass: directoryServerFeature
oid: 2.16.840.1.113730.3.4.9
cn: VLV Request Control
aci: (targetattr != "aci")(version 3.0; acl "VLV Request Control";
      allow( read, search, compare, proxy ) userdn = "ldap:///all" ;)
creatorsName: cn=server,cn=plugins,cn=config modifiersName:
cn=server,cn=plugins,cn=config ...
```

4. Change `"ldap:///all"` to `"ldap:///anyone"` and save your changes.

3. Deleting Indexes

This section describes how to delete presence, equality, approximate, substring, international, and browsing indexes for specific attributes.



NOTE

Because Directory Server 8.0 can operate in either a single or multi-database environment, you have to delete any unwanted indexes from every database instance. Any default indexes you delete will not be deleted from previous sets of

indexes on existing database instances.

- [Section 3.1, “Deleting Indexes from the Server Console”](#)
- [Section 3.2, “Deleting Indexes from the Command-Line”](#)
- [Section 3.3, “Deleting Browsing Indexes from the Server Console”](#)
- [Section 3.4, “Deleting Browsing Indexes from the Command-Line”](#)



CAUTION

Do *not* delete system indexes because deleting them can significantly affect Directory Server performance. System indexes are located in the `cn=index,cn=instance,cn=ldbm database,cn=plugins,cn=config` entry and the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

Also, be cautious when deleting default indexes since this can also affect how Directory Server works.

3.1. Deleting Indexes from the Server Console

The Directory Server Console allows you to delete any indexes you have created, indexes used by other server applications such as a messaging or web server, and default indexes.



NOTE

You cannot delete system indexes.

To delete indexes using the Directory Server Console, do the following:

1. Select the **Configuration** tab.
2. Expand the **Data** node and expand the suffix associated with the database containing the index.
3. Select the database from which to delete the index.

4. Locate the attribute containing the index to delete. Clear the checkbox under the index.

To delete all indexes maintained for a particular attribute, select the attribute's cell under **Attribute Name**, and click **Delete Attribute**.

5. Click **Save**.

A **Delete Index** warning dialog box opens, requiring a confirmation to delete the index.

6. Click **Yes** to delete the index.

7. The **Delete Browsing Index** dialog box appears displaying the status of the index deletion.

Click the **Status Logs** button to view the status of the indexes deleted.

8. Once the indexing is complete, click **Close**.

3.2. Deleting Indexes from the Command-Line

Creating browsing indexes, or virtual list view (VLV) indexes, using the `ldapdelete` command-line utility has two steps:

1. Delete an entire index entry or delete unwanted index types from an existing index entry using the `ldapdelete` command-line utility ([Section 3.2.1, "Deleting an Index Entry"](#)).
2. Generate the new set of indexes to be maintained by the server using the `db2index.pl` Perl script ([Section 3.2.2, "Running the db2index.pl Script"](#)).

3.2.1. Deleting an Index Entry

Use the `ldapdelete` command-line utility to delete either the entire indexing entry or the unwanted index types from an existing entry.

- To delete the indexes for a particular database, remove the index entry from the `cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=database_name` corresponds to the name of the database.
- To delete a default index, remove it from the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

1. Run `ldapdelete`.¹

```
ldapdelete -D "cn=Directory Manager" -w password -h ExampleServer
-p 389 "cn=sn,cn=index,cn=Example1,dn=ldbm
database,cn=plugins,dn=config"
```

- For complete information on `ldapdelete`, see the *Directory Server Configuration, Command, and File Reference*.
2. For example, delete the presence, equality, and substring indexes for the `sn` attribute on the database named `Example1`:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

Table 10.5, “*ldapdelete Options*” describes the `ldapdelete` options.

Option	Description
-D	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
-w	Specifies the password associated with the distinguished name specified in the <code>-D</code> option.
-h	Specifies the name of the host on which the server is running.
-p	Specifies the port number that the server uses.

Table 10.5. *Ldapdelete Options*

After deleting the index entry, the presence, equality, and substring indexes for the `sn` attribute are no longer maintained by the `Example1` database.

3.2.2. Running the `db2index.pl` Script

After deleting an indexing entry or some of the index types from an indexing entry, run the `db2index.pl` script to generate the new set of indexes to be maintained by the Directory Server. Once you run the script, the new set of indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `db2index.pl` Perl script, do the following:

1. Open the Directory Server instance directory.²

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Run the `db2index.pl` Perl script. For example:

```
db2index.pl -D "cn=Directory Manager" -w password -n Example1
```

For more information about using the `db2index.pl` Perl script, refer to *Directory Server Configuration, Command, and File Reference*. [Table 10.6, “db2index Options”](#) describes the `db2index.pl` options used in the examples:

Option	Description
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
-n	Specifies the name of the database into which you are importing the data.

Table 10.6. db2index Options

3.3. Deleting Browsing Indexes from the Server Console

To delete a browsing index through the Directory Server Console, do the following:

1. Select the **Directory** tab.
2. Select the entry from which to delete the index in the navigation tree, and select **Delete Browsing Index** from the **Object** menu.

Alternatively, select and right-click the entry of the index to delete in the navigation tree, and then choose **Delete Browsing Index** from the pop-up menu.
3. A **Delete Browsing Index** dialog box appears asking you to confirm the deletion of the index. Click **Yes**.
4. The **Delete Browsing Index** dialog box appears displaying the status of the index deletion.

3.4. Deleting Browsing Indexes from the Command-Line

Deleting a browsing index or virtual list view (VLV) index from the command-line involves two steps:

1. Using the `ldapdelete` to delete browsing index entries or edit existing browsing index entries ([Section 3.4.1, “Deleting a Browsing Index Entry”](#)).
2. Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server ([Section 3.4.2, “Running the vlvindex Script”](#)).

The actual entries for an alphabetical browsing index and virtual list view are the same. The following sections describe the steps involved in deleting browsing indexes.

3.4.1. Deleting a Browsing Index Entry

Use the `ldapdelete` command-line utility to either delete browsing indexing entries or edit existing browsing index entries. To delete browsing indexes for a particular database, remove the browsing index entries from the `cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=database_name` corresponds to the name of the database.

For example, there is a browsing index for accelerating `ldapsearch` operations on the entry `ou=People,dc=example,dc=com`. It held in the `Example1` database where the search base is `ou=People,dc=example,dc=com`, the search filter is `(|(objectclass=*)(objectclass=ldapsubentry))`, the scope is 1, and the sorting order for the returned attributes is `cn, givenName, o, ou, and sn`.

1. Run `ldapdelete`.¹

```
ldapdelete-D "cn=Directory Manager" -w password -h ExampleServer
-p 389 "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"
```

For full information on `ldapdelete` options, see the *Directory Server Configuration, Command, and File Reference*.

2. To delete this browsing index, delete the two corresponding browsing index entries:

```
dn: cn=MCC ou=People dc=example dc=com, cn=userRoot, cn=ldbm database,
cn=plugins, cn=config
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People, dc=example,dc=com
vlvScope: 1 vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))

dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
dc=example dc=com, cn=userRoot, cn=ldbm database, cn=plugins,cn=config
objectClass: top
objectClass: vlvIndex
```

```
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenName o ou sn
```

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

Option	Description
<code>-D</code>	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
<code>-w</code>	Specifies the password associated with the distinguished name specified in the <code>-D</code> option.
<code>-h</code>	Specifies the name of the host on which the server is running.
<code>-p</code>	Specifies the port number that the server uses.

After deleting the two browsing index entries, the browsing index will no longer be maintained by the `Example1` database.

3.4.2. Running the `vlvindex` Script

After deleting browsing indexing entries or unwanted attribute types from existing browsing indexing entries, run the `vlvindex` script to generate the new set of browsing indexes to be maintained by the Directory Server. After the script is run, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

1. Open the Directory Server instance directory.²

```
cd /usr/lib/dirsrv/slapd-instance_name
```

2. Stop the server.³

```
service dirsrv stop instance
```

3. Run the `vlvindex` script.

```
vlvindex -n Example1 -T "by MCC ou=people dc=example dc=com"
```

For more information about using the `vlvindex` script, see the *Directory Server Configuration, Command, and File Reference*.

4. Restart the server.

```
service dirsrv start instance
```

Table 10.4, “vlvindex Options” describes the `vlvindex` options.

4. Managing Indexes

Each index that the directory uses is composed of a table of index keys and matching entry ID lists. This entry ID list is used by the directory to build a list of candidate entries that may match a client application's search request; [Section 1, “About Indexes”](#) describes each kind of Directory Server index. The Directory Server secondary index structure greatly improves write and search operations.

4.1. Indexing Performance

While achieving extremely high read performance, in previous versions of Directory Server, write performance was limited by the number of bytes per second that could be written into the storage manager's transaction log file. Large log files were generated for each LDAP write operation; in fact, *log file verbosity* could easily be 100 times the corresponding number of bytes changed in the Directory Server. The majority of the contents in the log files are related to index changes (ID insert and delete operations).

The secondary index structure was separated into two levels in the old design:

- The ID list structures, which were the province of the Directory Server backend and opaque to the storage manager.
- The storage manager structures (Btrees), which were opaque to the Directory Server backend code.

Because it had no insight into the internal structure of the ID lists, the storage manager had to treat ID lists as opaque byte arrays. From the storage manager's perspective, when the content of an ID list changed, the *entire list* had changed. For a single ID that was inserted or deleted from an ID list, the corresponding number of bytes written to the transaction log was the maximum configured size for that ID list, about 8 kilobytes. Also, every database page on which the list was stored was marked as dirty, since the *entire* list had changed.

In the redesigned index, the storage manager has visibility into the fine-grain index structure, which optimizes transaction logging so that only the number of bytes actually changed need to be logged for any given index modification. The Berkeley DB provides ID list semantics, which are implemented by the storage manager. The Berkeley API was enhanced to support the

insertion and deletion of individual IDs stored against a common key, with support for duplicate keys, and an optimized mechanism for the retrieval of the complete ID list for a given key.

The storage manager has direct knowledge of the application's intent when changes are made to ID lists, resulting in several improvements to ID list handling:

- For long ID lists, the number of bytes written to the transaction log for any update to the list is significantly reduced, from the maximum ID list size (8 kilobytes) to twice the size of one ID (4 bytes).
- For short ID lists, storage efficiency, and in most cases performance, is improved because only the storage manager metadata need to be stored, not the ID list metadata.
- The average number of database pages marked as dirty per ID insert or delete operation is very small because a large number of duplicate keys will fit into each database page.

4.2. Search Performance

For each entry ID list, there is a size limit that is globally applied to all index keys managed by the server. In previous versions of Directory Server, this limit was called the *All IDs Threshold*. Because maintaining large ID lists in memory can affect performance, the All IDs Threshold set a limit on how large a single entry ID list could get. When a list hit a certain pre-determined size, the search would treat it as if the index contained the entire directory.

The difficulty in setting the All IDs Threshold hurt performance. If the threshold was too low, too many searches examined every entry in the directory. If it was too high, too many large ID lists had to be maintained in memory.

The problems addressed by the All IDs Threshold are no longer present because of the efficiency of entry insertion, modification, and deletion in the Berkeley DB design. The All IDs Threshold is removed for database write operations, and every ID list is now maintained accurately.

Since loading a long ID list from the database can significantly reduce search performance, the configuration parameter, `nsslapd-idlistscanlimit`, sets a limit on the number of IDs that are read before a key is considered to match the entire primary index. `nsslapd-idlistscanlimit` is analogous to the All IDs Threshold, but it only applies to the behavior of the server's search code, not the content of the database.

When the server uses indexes in the processing of a search operation, it is possible that one index key matches a large number of entries. For example, consider a search for `objectclass=inetorgperson` in a directory that contained one million `inetorgperson` entries. When the server reads the `inetorgperson` index key in the `objectclass` index, it finds one million matching entries. In cases like this, it is more efficient simply to conclude in the index lookup phase of the search operation processing that all the entries in the database match the query. This causes subsequent search processing to scan the entire database content, checking each entry as to whether it matches the search filter. The time required to fetch the

index keys is not worthwhile; the search operation is likely to be processed more efficiently by omitting the index lookup.

In Directory Server, when examining an index, if more than a certain number of entries are found, the server stops reading the index and marks the search as unindexed for that particular index.

The threshold number of entries is called the `idlistscanlimit` and is configured with the `nsslapd-idlistscanlimit` configuration attribute. The default value is 4000, which is designed to give good performance for a common range of database sizes and access patterns. Typically, it is not necessary to change this value. However, in rare circumstances it may be possible to improve search performance with a different value. For example, lowering the value will improve performance for searches that will otherwise eventually hit the default limit of 4000. This might reduce performance for other searches that benefit from indexing. Conversely, increasing the limit could improve performance for searches that were previously hitting the limit. With a higher limit, these searches could benefit from indexing where previously they did not.

For more information on search limits for the server, refer to [Section 1.3, “Overview of the Searching Algorithm”](#).

4.3. Backwards Compatibility and Migration

While current versions of Directory Server can support the old database design, only the new design is supported for this and later releases of Directory Server.

Upon startup, the server will read the database version from the `DBVERSION` file, which contains the text `Netscape-ldbm/6.2` (old database version), `Netscape-ldbm/7.1` (new database format), or `bdb/4.2/libback-ldbm` (new database format). If the file indicates that the old format is used, then the old code is selected for the database. Because the `DBVERSION` file stores everything per-backend, it is possible to have different database formats for different individual backends, but the old database format is not recommended.

All databases must be migrated to Directory Server 8.0 when the system is upgraded. Migration is supported for Directory Server 6.x versions, and for releases earlier than version 6.x, dump the databases be dumped, and install Directory Server fresh. Migrating databases is covered in the *Directory Server Installation Guide*.

Also, the index sizes can be larger than in older releases, so you may want to increase your database cache size. To reconfigure your cache size, look up the `nsslapd-dbcachesize` entry in the *Directory Server Configuration, Command, and File Reference*.

5. Attribute Name Quick Reference Table

[Table 10.7, “Attribute Name Quick Reference Table”](#) lists all attributes which have a primary or real name as well as an alias. When creating indexes be sure to use the primary name.

Attribute Primary Name	Attribute Alias
dn	distinguishedName

Attribute Primary Name	Attribute Alias
cn	commonName
sn	surName
c	countryName
l	localityName
st	stateOrProvinceName
street	streetAddress
o	organization
ou	organizationalUnitName
facsimileTelephoneNumber	fax
uid	userId
mail	rfc822mailbox
mobile	mobileTelephoneNumber
pager	pagerTelephoneNumber
co	friendlyCountryName
labeledUri	labeledUri
ttl	timeToLive
dc	domainComponent
authorCn	documentAuthorCommonName
authorSn	documentAuthorSurname
drink	favoriteDrink

Table 10.7. Attribute Name Quick Reference Table

Managing SSL

To provide secure communications over the network, Red Hat Directory Server includes the LDAPS communications protocol. LDAPS is the standard LDAP protocol, running over Transport Layer Security (TLS, formerly Secure Sockets Layer or SSL). Directory Server also allows *spontaneous* secure connections over otherwise-insecure LDAP ports, using the Start TLS LDAP extended operation.

This chapter describes how to use SSL with Directory Server.

1. Introduction to SSL in the Directory Server

The Directory Server supports TLS/SSL to secure communications between LDAP clients and the Directory Server, between Directory Servers that are bound by a replication agreement, or between a database link and a remote database. Directory Server can use TLS/SSL with simple authentication (bind DN and password) or with certificate-based authentication.

Directory Server's cryptographic services are provided by Mozilla Network Security Services (NSS), a library of TLS/SSL and base cryptographic functions. NSS includes a software-based cryptographic token which is FIPS 140-2 certified.

Using TLS/SSL with simple authentication ensures confidentiality and data integrity. There are two major benefits to using a certificate — smart card, token, or software-based — to authenticate to the Directory Server instead of a bind DN and password:

- *Improved efficiency.* When using applications that prompt once for the certificate database password and then use that certificate for all subsequent bind or authentication operations, it is more efficient than continuously providing a bind DN and password.
- *Improved security.* The use of certificate-based authentication is more secure than non-certificate bind operations because certificate-based authentication uses public-key cryptography. Bind credentials cannot be intercepted across the network. If the certificate or device is lost, it is useless without the PIN, so it is immune from third-party interference like phishing attacks.

The Directory Server is capable of simultaneous TLS/SSL and non-SSL communications. This means that you do not have to choose between TLS/SSL or non-SSL communications for the Directory Server; both can be used at the same time. Directory Server can also utilize the Start TLS extended operation to allow TLS/SSL secure communication over a regular (insecure) LDAP port.

1.1. Enabling SSL: Summary of Steps

To configure the Directory Server to use LDAPS, follow these steps:

1. Obtain and install a certificate for the Directory Server, and configure the Directory Server to trust the certification authority's (CA's) certificate.

For information, see [Section 2, “Obtaining and Installing Server Certificates”](#).

2. Turn on SSL in the directory.

For information, refer to [Section 4, “Starting the Server with SSL Enabled”](#).

3. Configure the Administration Server connect to an SSL-enabled Directory Server.

4. Optionally, ensure that each user of the Directory Server obtains and installs a personal certificate for all clients that will authenticate with TLS/SSL.

For information, refer to [Section 7, “Configuring LDAP Clients to Use SSL”](#).

1.2. Command-Line Functions for Start TLS

LDAP operations such as `ldapmodify`, `ldapssearch`, and `ldapdelete` can use TLS/SSL when communicating with an SSL-enabled server or to use certificate authentication. Command-line options also specify or enforce Start TLS, which allows a secure connection to be enabled on a clear text port after a session has been initiated.



IMPORTANT

These options to use Start TLS applies only for the Mozilla LDAP tools provided with Red Hat Directory Server.

In the following example, a network administrator enforces Start TLS for a search for Mike Connor's identification number:

```
ldapssearch -p 389 -ZZZ -P certificateDB -s base  
-b "uid=mconnors,ou=people,dc=example,dc=com" "(attribute=govIdNumber)"
```

`-zzz` enforces Start TLS, and `certificateDB` gives the filename and path to the certificate database.



NOTE

The `-zzz` option enforces the use of Start TLS, and the server must respond that a Start TLS command was successful. If the `-zzz` command is used and the server does not support Start TLS, the operation is aborted immediately.

For information on the command-line options available, see the *Directory Server Configuration*,

Command, and File Reference.

1.2.1. Troubleshooting Start TLS

With the `-zz` option, the following errors could occur:

- If there is no certificate database, the operation fails. See [Section 2, “Obtaining and Installing Server Certificates”](#) for information on using certificates.
- If the server does not support Start TLS, the connection proceeds in clear text. To enforce the use of Start TLS, use the `-zzz` command option.
- If the certificate database does not have the certificate authority (CA) certificate, the connection proceeds in clear text. See [Section 2, “Obtaining and Installing Server Certificates”](#) for information on using certificates.

With the `-zzz` option, the following errors could occur, causing the Start TLS operation to fail:

- If there is no certificate database. See [Section 2, “Obtaining and Installing Server Certificates”](#) for information on using certificates.
- If the certificate database does not have the certificate authority (CA) certificate. See [Section 2, “Obtaining and Installing Server Certificates”](#) for information on using certificates.
- The server does not support Start TLS as an extended operation.

For SDK libraries used in client programs, if a session is already in TLS mode and Start TLS is requested, then the connection continues to be in secure mode but prints the error `"DSA is unwilling to perform"`.

2. Obtaining and Installing Server Certificates

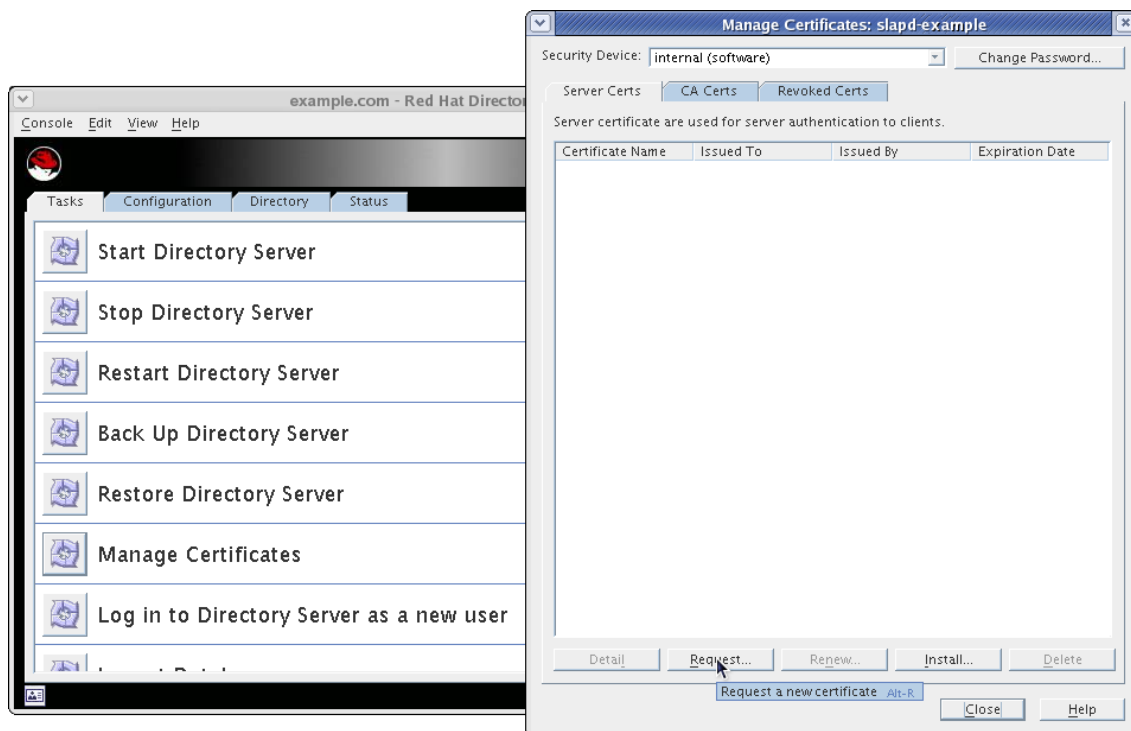
Before the Directory Server can be set to run in TLS/SSL, server and CA certificates must be properly configured in the Directory Server. If a server certificate has already been generated for the Directory Server instance and the issuing certificate authority (CA) is already trusted by the Directory Server, begin setting up TLS/SSL as described in [Section 4, “Starting the Server with SSL Enabled”](#).

Obtaining and installing certificates consists of the following steps:

1. Generate a certificate request.
2. Send the certificate request to a certificate authority.
3. Install the server certificate.

4. Set the Directory Server to trust the certificate authority.
5. Confirm that the certificates are installed.

Two wizards automate the process of creating a certificate database and of installing the key-pair. The **Certificate Request Wizard** in the Directory Server Console can generate a certificate request and send it to a certificate authority. The **Certificate Install Wizard** in the Directory Server Console can then install the server certificate and the CA certificate.



2.1. Step 1: Generate a Certificate Request

Generate a certificate request, and send it to a CA. The Directory Server Console has a tool, the **Certificate Request Wizard**, which generates a valid certificate request to submit to any certificate authority (CA).

1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
2. Select the **Server Certs** tab, and click the **Request** button. This opens the **Certificate Request Wizard**.
3. Click **Next**.
4. Enter the **Requester Information** in the blank text fields, then click **Next**.

The image shows a Windows-style dialog box titled "Certificate Request Wizard" with a close button in the top right corner. The dialog is divided into two panes, with the right pane showing "2 of 4". The left pane is titled "Requestor Information". It contains several input fields: "Server name:" with the text "example-server" and a cursor; "Organization:" with the text "Example Corp."; "Organizational unit:" with the text "Engineering"; "City/locality:" with the text "Raleigh"; "State/province:" with a dropdown menu showing "North Carolina"; and "Country/region:" with a dropdown menu showing "US United States". At the bottom right of the dialog is a "Show DN" button. At the bottom center are four buttons: "< Back", "Next >", "Cancel", and "Help".

- *Server Name.* Enter the fully qualified hostname of the Directory Server as it is used in DNS and reverse DNS lookups; for example, `dir.example.com`. The server name is critical for client-side validation to work, which prevents man-in-the-middle attacks.
 - *Organization.* Enter the legal name of the company or institution. Most CAs require this information to be verified with legal documents such as a copy of a business license.
 - *Organizational Unit. Optional.* Enter a descriptive name for the organization within the company.
 - *Locality. Optional.* Enter the company's city name.
 - *State or Province.* Enter the full name of the company's state or province (no abbreviations).
 - *Country.* Select the two-character abbreviation for the country's name (ISO format). The country code for the United States is US.
5. Enter the password that will be used to protect the private key, and click **Next**.



The **Next** button is grayed out until a password is supplied.

6. The **Request Submission** dialog box provides two ways to submit a request: directly to the CA (if there is one internally) or manually. To submit the request manually, select **Copy to Clipboard** or **Save to File** to save the certificate request which will be submitted to the CA.



7. Click **Done** to dismiss the **Certificate Request Wizard**.

After generating the certificate request, send it to the CA.

2.2. Step 2: Send the Certificate Request

After the certificate request is generated, send it to a certificate authority (CA); the CA will generate return a server certificate.

1. Most certificate requests are emailed to the CA, so open a new message.
2. Copy the certificate request information from the clipboard or the saved file into the body of the message.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCVXMxEzARBgNVBAGTCkNBTElGT1J
OSUEXLDAqBgVBaoTI25ldHNjYXB1IGNvbW11bmljYXRpb25zIGNvcnBvcnF
0aW9uMRwwGgYDVQQDExNtZWxs24ubmV0c2NhG0UuY29tMIGfMA0GCSqGSI
b3DQEBAQUAA4GNADCBiQKBgQCWAbskGh6SKYOGHy+UCSLnm3ok3X3u83Us7
ug0EfgSLR0f+K41eNqgRftGR83emqPLDof0ZLTLjVGJaH4Jn411gG+JDf/n
/zMyahxtv7+mT8GOFFigFfuxaxMjr2j7IvELlxQ4IfZgWwqCm4qQecv3G+N
9YdbjveMVXW0v4XwIDAQABoAAwDQYK
-----END NEW CERTIFICATE REQUEST-----
```

3. Send the email message to the CA.

After emailing the certificate request, wait for the CA to respond with the server certificate. Response time for requests varies. For example, if the CA is internal to the company, it may only take a day or two to respond to the request. If the selected CA is a third-party, it could take several weeks to respond to the request.

After receiving the certificate, install it in the Directory Server's certificate database. When the CA sends a response, be sure to save the information in a text file. The certificate must be available to install in the Directory Server.

Also, keep a backup of the certificate data in a safe location. If the system ever loses the certificate data, the certificate can be reinstalled using the backup file.

2.3. Step 3: Install the Certificate

1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
2. Select the **Server Certs** tab, and click **Install**.
3. Give the certificate location or paste the certificate text in the text box, then click **Next**.

Certificate Install Wizard 1 of 4

Where is the certificate you want to install located?

☐ to be issued by this CA:

☐ in this local file:

☒ in the following encoded text block:

```
-----BEGIN CERTIFICATE-----
MIICMjCCAZugAwMBAglCCEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMx
IzAhBgNVBAoTGIBhbG9va2FWaWxsZSBXaWRnZXRzLCBjbmuMR0wGwYDVQQLEXRx
aWRnZXQgTWFrZXJzIEdSjyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVz
dCBUZXN0IFRlc3QgQ0EwHhcNOTgwwMzEyMDIzMzU3WhcNOTgwwMzI2MDIzMzU3WjBP
MQswCQYDVQQGEwJlUzEoMCYGA1UEChMfTmV0c2NhcnUgRGlyZWN0b3J5IFB1YmVp
Y2F0aW9ucyEwMBQGA1UEAxMNZHVhZ9dq2itLmNvbTBaMA0GCSqGSIb3
-----END CERTIFICATE-----
```

- *In this file.* Enter the absolute path to the certificate in this field.
- *In the following encoded text block.* Copy the text from the CA's email or from the created

text file, and paste it in this field.

4. Check that the certificate information displayed is correct, and click **Next**.
5. Give a name to the certificate, and click **Next**.
6. Provide the password that protects the private key. This password is the same as the one provided in step 5 in [Section 2.1](#), “[Step 1: Generate a Certificate Request](#)”.

After installing the server certificate, configure the Directory Server to trust the CA which issued the server's certificate.

2.4. Step 4: Trust the Certificate Authority

Configuring the Directory Server to trust the certificate authority consists of obtaining the CA's certificate and installing it into the server's certificate database. This process differs depending on the certificate authority. Some commercial CAs provide a web site that allow users to automatically download the certificate. Others will email it back to users.

After receiving the CA certificate, use the **Certificate Install Wizard** to configure the Directory Server to trust the certificate authority.

1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
2. Go to the **CA Certs** tab, and click **Install**.
3. If the CA's certificate is saved to a file, enter the path in the field provided. Alternatively, copy and paste the certificate, including the headers, into the text box. Click **Next**.
4. Check that the certificate information that opens is correct, and click **Next**.
5. Name the certificate, and click **Next**.
6. Select the purpose of trusting this certificate authority; it is possible to select both options:
 - *Accepting connections from clients (Client Authentication)*. The server checks that the client's certificate has been issued by a trusted certificate authority.
 - *Accepting connections to other servers (Server Authentication)*. This server checks that the directory to which it is making a connection (for replication updates, for example) has a certificate that has been issued by a trusted certificate authority.
7. Click **Done**.

Once both the server and CA certificates are installed, it is possible to configure the Directory Server to run in TLS/SSL. However, Red Hat recommends verifying that the certificates have been installed correctly.

2.5. Step 5: Confirm That The New Certificates Are Installed

1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
2. Select the **Server Certs** tab.

A list of all the installed certificates for the server opens.

3. Scroll through the list. The certificates installed previously should be listed.

It is now possible to set up the Directory Server to run in TLS/SSL.



NOTE

When renewing a certificate using the **Certificate Wizard**, the text on the introduction screen does not clearly indicate that the process is renewal and not requesting a new certificate. Also, the requester information is not filled in automatically.

3. Using certutil

The Directory Server has a command-line tool, `certutil`, which locally creates self-signed CA and client certificates, certificate databases, and keys. The default location for the Directory Server `certutil` tool is `/usr/lib/dirsec/`.¹



TIP

Set the environment variable for the shell to include the `certutil` directory path. For example:

```
export PATH=/usr/lib/dirsec/:$PATH
```

The command varies depending on the shell.

`certutil` can also be downloaded from <ftp://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/>.

3.1. Creating Directory Server Certificates through the Command Line

¹ This is the location for Red Hat Enterprise Linux 5 i386. File locations for other platforms are listed in [Section 1, "Directory Server File Locations"](#).

The following steps outline how to make the databases, key, CA certificate, server/client certificate, and convert the certificates into `pkcs12` format.

1. Open the directory where the Directory Server certificate databases are stored. For example:

```
cd /etc/dirsrv/slaped-instance_name
```

2. Create a temporary working directory, and open that directory.

```
mkdir /tmp  
cd /tmp
```

3. Create a password file for the security token password:

```
vi pwdfilere.txt  
  
secretpw
```

The password file should be owned by the user as which Directory Server runs, by default `nobody`, and it must be set as read-only for the Directory Server user and allow no access to anyone else (mode `0400`).

4. Create a noise file for the encryption mechanism:

```
vi noise.txt  
  
dsadasdasdasdasdasdasdasdsadferwerjfdksdjfkdsdlfhjsdk
```

5. Create the key and certificate databases databases.

```
certutil -N -d . -f pwdfilere.txt
```

6. Generate the encryption key:

```
certutil -G -d . -z noise.txt -f pwdfilere.txt
```

7. Generate the self-signed CA certificate. This certificate is used to generate the other server certificates and can be exported for use with other servers and clients.

```
certutil -S -n "CA certificate" -s "cn=CAcert" -x -t "CT,,"  
-m 1000 -v 120 -d . -z noise.txt -f pwdfilere.txt
```

8. Generate the Directory Server client certificate.

```
certutil -S -n "Server-Cert" -s "cn=FQDN,cn=Directory Server" -c "CA
certificate"
-t "u,u,u" -m 1001 -v 120 -d . -z noise.txt -f pwdfile.txt
```

FQDN is the fully-qualified host and domain name of the Directory Server, such as `ldap.example.com`. This name must be available for DNS and reverse DNS lookups to Directory Server clients because certificate validation may fail if the clients cannot properly resolve the *FQDN*. To use the Directory Server behind a DNS round robin or any other scheme which aliases a single server certificate to multiple hostnames, see the SSL information about server name wildcards or `subjectAltName`.

To generate a client certificate to use with applications other than the Directory Server, run the same command as for the Directory Server certificate.



NOTE

Keep careful track on the numbers set with the `-m` option. The `-m` option sets the unique identifier for the server certificate, and a CA cannot issue two certificates with the same ID.

9. Move the new key and certificate databases and copy over the default Directory Server databases.

```
mv .. # If the /tmp directory is in /etc/dirsrv/slapd-instance_name
```

10. Export the CA certificate to ASCII (PEM) format so it can be used with other clients.

```
certutil -L -d . -n "CA certificate" -a > cacert.asc
```

The `cacert.asc` file can be used as the CA certificate for most clients that require the CA certificate in a simple ASCII file, including Red Hat Enterprise Linux clients, web servers, and others.

11. Optionally, run `pk12util` to create a `pkcs12` file of the CA certificate and Directory Server key and certificate as a backup. The `pkcs12` file contains sensitive information, so the file is password-protected and prompts for a password as the command runs.

```
pk12util -d . -o cacert.pk12 -n "CA certificate"
pk12util -d . -o dscert.pk12 -n "Server-Cert"
```

The certificates created by `certutil` are automatically available in the **Encryption** tab of the

Console; there is no need to import them.

3.2. certutil Usage

`certutil` can be used for a variety of tasks to manage certificates and keys, such as generating certificate requests and removing certificates from the certificate database. Some of the most common options are listed in [Table 11.1, “certutil Options”](#). For the full list of commands and arguments, run `certutil -H` from the command line.

certutil Options	Description
<code>certutil -L -d .</code>	Lists the certificates in the database.
<code>certutil -L -d . -n "cert_name"</code>	"Pretty prints" the specified certificate; the <i>cert_name</i> can specify either a CA certificate or a client certificate.
<code>certutil -L -d . -n "cert_name" > certfile.asc</code>	Exports the specified certificate out of the database to ASCII (PEM) format.
<code>certutil -L -d . -n "cert_name" -r > certfile.bin</code>	Exports the specified certificate out of the database to binary format; this can be used with Directory Server attributes such as <i>userCertificate;binary</i> .

Table 11.1. certutil Options

4. Starting the Server with SSL Enabled

Most of the time, the server should run with SSL enabled. If SSL is temporarily disabled, re-enable it before processing transactions that require confidentiality, authentication, or data integrity.

Before TLS/SSL can be activated, first create a certificate database, obtain and install a server certificate, and trust the CA's certificate, as described in [Section 2, “Obtaining and Installing Server Certificates”](#).

With TLS/SSL enabled, when the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database. Restarting the Directory Server without the password prompt is possible by using use a hardware crypto device or creating a PIN file ([Section 4.3, “Creating a Password File for the Directory Server”](#)).



NOTE

On SSL-enabled servers, be sure to check the file permissions on certificate database files, key database files, and PIN files to protect the sensitive information they contain. Because the server does not enforce read-only

permissions on these files, check the file modes to protect the sensitive information contained in these files.

The files must be owned by the Directory Server user, such as the default `nobody`. The key and cert databases should be owned by the Directory Server user and should typically have read/write access for the owner with no access allowed to any other user (mode `0600`). The PIN file should also be owned by the Directory Server user and set to read-only by this user, with no access to anyone other user (mode `0400`).

4.1. Enabling SSL Only in the Directory Server

1. Obtain and install CA and server certificates.
2. Set the secure port for the server to use for TLS/SSL communications.

The encrypted port number *must not* be the same port number used for normal LDAP communications. By default, the standard port number is 389, and the secure port is 636.

- a. Change the secure port number in the **Configuration>Settings** tab of the Directory Server Console.
 - b. Restart the Directory Server. It restarts over the regular port.
3. In the Directory Server Console, select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane. Select the **Encryption** tab in the right pane.
 4. Select the **Enable SSL for this Server** checkbox.
 5. Check the **Use this Cipher Family** checkbox.
 6. Select the certificate to use from the drop-down menu.
 7. Click **Cipher Settings**.

The **Cipher Preference** dialog box opens. By default, all ciphers are selected.

8. Set the preferences for client authentication.
 - *Do not allow client authentication.* With this option, the server ignores the client's certificate. This does not mean that the bind will fail.
 - *Allow client authentication.* This is the default setting. With this option, authentication is performed on the client's request. For more information about certificate-based authentication, see [Section 6, "Using Certificate-Based Authentication"](#).

- *Require client authentication.* With this option, the server requests authentication from the client.

If TLS/SSL is only enabled in the Directory Server and not the Directory Server Console, do not select **Require client authentication** checkbox.



NOTE

To use certificate-based authentication with replication, the consumer server must be configured either to allow or to require client authentication.

9. To verify the authenticity of requests, select the **Check hostname against name in certificate for outbound SSL connections** option. The server does this verification by matching the hostname against the value assigned to the common name (`cn`) attribute of the subject name in the being presented for authentication.

By default, this feature is disabled. If it's enabled and if the hostname does not match the `cn` attribute of the certificate, appropriate error and audit messages are logged. For example, in a replicated environment, messages similar to these are logged in the supplier server's log files if it finds that the peer server's hostname doesn't match the name specified in its certificate:

```
[DATE] - SSL alert: ldap_sasl_bind("",LDAP_SASL_EXTERNAL) 81 (Netscape
runtime error -12276 -
    Unable to communicate securely with peer: requested domain name does not
match the server's
    certificate.)
[DATE] NSMMReplicationPlugin - agmt="cn=to ultra60 client auth"
(ultra60:1924): Replication
    bind with SSL client authentication failed: LDAP error 81 (Can't contact
LDAP server)
```

Red Hat recommends enabling this option to protect Directory Server's outbound SSL connections against a man-in-the-middle (MITM) attack.

10. Click **Save**.

11. Restart the Directory Server. The Directory Server must be restarted from the command line.

²

```
service dirsrv restart instance
```

² The commands to start, stop, and restart the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, "Starting and Stopping Servers"](#).

When the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database.

To restart the Directory Server without the password prompt, create a PIN file or use a hardware crypto device. See [Section 4.3, “Creating a Password File for the Directory Server”](#) for information on how to create a PIN file.

4.2. Enabling SSL in the Directory Server, Administration Server, and Console

1. Obtain server certificates and CA certs, and install them on the Directory Server. This is described in [Section 2, “Obtaining and Installing Server Certificates”](#).
2. Obtain and install server and CA certificates on the Administration Server. This is a similar process as for the Directory Server.



NOTE

It is important that the Administration Server and Directory Server have a CA certificate in common so that they can trust the other's certificates.

3. If the default port number of 636 is not used, change the secure port setting.
 - a. Change the secure port number in the **Configuration>Settings** tab of the Directory Server Console, and save.
 - b. Restart the Directory Server. It restarts over the regular port. ²

```
service dirsrv restart instance
```

4. In the **Configuration** tab of the Directory Server Console, highlight the server name at the top of the table, and select the **Encryption** tab.
5. Select the **Enable SSL** checkbox.
6. Check the **Use this Cipher Family** checkbox.
7. Select the certificate to use from the drop-down menu.
8. Click **Cipher Settings**.

The **Cipher Preference** dialog box opens. By default, all ciphers are selected.

9. Set the preferences for client authentication.

- *Do not allow client authentication.* With this option, the server ignores the client's certificate. This does not mean that the bind will fail.
- *Allow client authentication.* This is the default setting. With this option, authentication is performed on the client's request. For more information about certificate-based authentication, see [Section 6, "Using Certificate-Based Authentication"](#).
- *Require client authentication.* With this option, the server requests authentication from the client.



NOTE

To use certificate-based authentication with replication, then configure the consumer server either to allow or to require client authentication.

10. To verify the authenticity of requests, select the **Check hostname against name in certificate for outbound SSL connections** option. The server does this verification by matching the hostname against the value assigned to the common name (`cn`) attribute of the subject name in the being presented for authentication.

By default, this feature is disabled. If it's enabled and if the hostname does not match the `cn` attribute of the certificate, appropriate error and audit messages are logged. For example, in a replicated environment, messages similar to these are logged in the supplier server's log files if it finds that the peer server's hostname doesn't match the name specified in its certificate:

```
[DATE] - SSL alert: ldap_sasl_bind("",LDAP_SASL_EXTERNAL) 81 (Netscape
runtime error -12276 -
    Unable to communicate securely with peer: requested domain name does not
match the server's
    certificate.)
[DATE] NSMMReplicationPlugin - agmt="cn=to ultra60 client auth"
(ultra60:1924): Replication
    bind with SSL client authentication failed: LDAP error 81 (Can't contact
DAP server)
```

Red Hat recommends enabling this option to protect Directory Server's outbound SSL connections against a man-in-the-middle (MITM) attack.

11. Check the **Use SSL in the Console** box. Hit **Save**.

12. In the Administration Server Console, select the **Configuration** tab. Select the **Encryption** tab, check the **Enable SSL** checkbox, and fill in the appropriate certificate information.

13. In the **Configuration DS** tab, change the port number to the new Directory Server secure

port information. See [Section 5, “Changing Directory Server Port Numbers”](#) for more information. Do this even if the default port of 636 is used. Check the **Secure Connection** checkbox.

14 In the **User DS** tab, select the **Set User Directory** radio button, and fill in the Directory Server secure port information, the LDAP URL, and the user database information. Check the **Secure Connection** checkbox.

15 Save the new SSL settings and **Configuration DS** and **User DS** information in the Administration Server Console.

16 Restart the Directory Server. The server must be restarted from the command line.²

```
service dirsrv restart instance
```

When the server restarts, it prompts for the PIN or password to unlock the key database. This is the same password used when the server certificate and key were imported into the database.

To restart the Directory Server without the password prompt, create a PIN file or use a hardware crypto device. See [Section 4.3, “Creating a Password File for the Directory Server”](#) for information on how to create a PIN file.



NOTE

When next logging into the Directory Server Console, be certain that the address reads `https`; otherwise, the operation will time out, unable to find the server since it is running on a secure connection. After successfully connecting, a dialog box appears to accept the certificate. Click **OK** to accept the certificate (either only for that current session or permanently).

4.3. Creating a Password File for the Directory Server

It is possible to store the certificate password in a password file. By placing the certificate database password in a file, the server can be started from the Directory Server Console and also restarted automatically when running unattended.



CAUTION

This password is stored in clear text within the password file, so its usage represents a significant security risk. Do not use a password file if the server is running in an unsecured environment.

The password file must be in the same directory where the other key and certificate databases for Directory Server are stored. This is usually the main configuration directory, `/etc/dirsrv/slapd-instance_name`. The file should be named `pin.txt`.

Include the token name and password in the file, such as *Token:mypassword*. For example:

```
Internal (Software) Token:mypassword
```

For the NSS software crypto module, the token is always called "Internal (Software) Token".

The PIN file should be owned by the Directory Server user and set to read-only by the Directory Server user, with no access to anyone other user (mode `0400`).

4.4. Creating a Password File for the Administration Server

Like the Directory Server, the Administration Server can use a password file during login when SSL is enabled.



CAUTION

This password is stored in clear text within the password file, so its usage represents a significant security risk. Do not use a password file if the server is running in an unsecured environment.

1. Open the Administration Server configuration directory, `/etc/dirsrv/admin-serv`.
2. Create a password file named `password.conf`. The file should include a line with the token name and password, in the form *Token:mypassword*. For example:

```
Internal (Software) Token:mypassword
```

The password file should be owned by the Administration Server user and set to read-only by the Administration Server user, with no access to any other user (mode `0400`).



TIP

To find out what the Administration Server user ID is, run `grep` in the Administration Server configuration directory:

```
cd /etc/dirsrv/admin-serv
```

```
grep ^User console.conf
```

3. In the `/etc/dirsrv/admin-serv` directory, edit the `nss.conf` file to point to the location of the new password file.

```
# Pass Phrase Dialog:
# Configure the pass phrase gathering process.
# The filtering dialog program ('builtin' is a internal
# terminal dialog) has to provide the pass phrase on stdout.
NSSPassPhraseDialog file:///etc/dirsrv/admin-serv/password.conf
```

4. Restart the Administration Server. ²

```
service dirsrv-admin restart
```

5. Setting Security Preferences

The Directory Server supported several different ciphers, and the type of ciphers to use for TLS/SSL communications are set by the user. A *cipher* is the algorithm used in encryption. Some ciphers are more secure, or stronger, than others. Generally speaking, the more bits a cipher uses during encryption, the more difficult it is to decrypt the key.

When a client initiates an TLS/SSL connection with a server, the client tells the server what ciphers it prefers to use to encrypt information. In any two-way encryption process, both parties must use the same ciphers. There are a number of ciphers available. The server needs to be able to use the ciphers that will be used by client applications connecting to the server.

5.1. Available Ciphers

This section lists information about the available ciphers for Directory Server encryption. Each cipher has the following information:

- *Directory Server name.* The name of the cipher suite used when configuring the Directory Server. The Directory Server uses this name both internally and in the Directory Server Console.
- *Key exchange.* The key exchange algorithm. DHE stands for Diffie-Hellman; DSS stands for Digital Signature Standard. The 1024 bit ciphers are lower strength ciphers formerly used for export control.

- *Encryption Algorithm.* AES stands for the American Encryption Standard. DES stands for Data Encryption Standard.
- *Symmetric Key Bit Size.* The size in bits of the key used for the actual transport data encryption.
- *Message Authentication.* SHA stands for Secure Hash Algorithm.

The Mozilla site,

<http://www.mozilla.org/projects/security/pki/nss/nss-3.11/nss-3.11-algorithms.html> for definitions and explanations of the encryption algorithms.



NOTE

Directory Server supports ciphers for TLSv1 (recommended) and SSLv3. SSLv2 support is deprecated and not enabled by default in Directory Server.

Directory Server provides the following TLSv1 ciphers:

Directory Server Name	Key Exchange	Encryption Algorithm	Symmetric Key Bit Size	Message Authentication
tls_dhe_dss_aes_128_sha	DHE with DSS	AES	128	SHA
tls_dhe_rsa_aes_128_sha	DHE with RSA	AES	128	SHA
tls_rsa_aes_256_sha	RSA	AES	256	SHA
tls_dhe_dss_aes_256_sha	DHE with DSS	AES	256	SHA
tls_dhe_rsa_aes_256_sha	DHE with RSA	AES	256	SHA
tls_dhe_dss_rc4_1024_sha	DHE with DSS 1024 bit public key	RC4	56	SHA
tls_dhe_dss_rc4_128_sha	DHE with DSS	RC4	128	SHA
tls_rsa_export1024_rc4_1024_sha	RSA with 1024 bit public key	RC4	56	SHA
tls_rsa_export1024_rc4_128_sha	RSA with 1024 bit public key	DES	56	SHA

Table 11.2. TLSv1 Ciphers

Directory Server provides the following SSLv3 ciphers:

Directory Server Name	Key Exchange	Encryption Algorithm	Symmetric Key Bit Size	Message Authentication
dhe_rsa_3des_sha	DHE with RSA	3DES	168	SHA

Directory Server Name	Key Exchange	Encryption Algorithm	Symmetric Key Bit Size	Message Authentication
dhe_rsa_des_sha	DHE with RSA	DES	56	SHA
dhe_dss_3des_sha	DHE with DSS	3DES	168	SHA
dhe_dss_des_sha	DHE with DSS	DES	56	SHA
rsa_des_sha	RSA	DES	56	SHA
rsa_3des_sha	RSA	3DES	168	SHA
rsa_fips_des_sha	RSA	DES	56	SHA
rsa_fips_3des_sha	RSA	3DES	168	SHA
rsa_rc4_128_md5	RSA	RC4	128	MD5
rsa_rc4_40_md5	RSA	RC4	40	MD5
rsa_rc2_40_md5	RSA	RC2	40	MD5
rsa_null_md5	RSA	null (none)	N/A	MD5
fortezza	fortezza	fortezza	80	SHA
fortezza_rc4_128_md5	fortezza	RC4	128	SHA
fortezza_null	fortezza	null (none)	N/A	SHA

Table 11.3. SSLv3 Ciphers

5.2. Selecting the Encryption Cipher

To select the ciphers for the Directory Server to use, do the following:

1. Make sure TLS/SSL is enabled for the server. For instructions on enabling TLS/SSL, see [Section 4, “Starting the Server with SSL Enabled”](#).
2. In the Directory Server Console, select the **Configuration** tab, and then select the topmost entry in the navigation tree in the left pane.
3. Select the **Encryption** tab in the right pane.

This displays the current server encryption settings.

4. Click **Cipher Setting**.

The **Cipher Preference** dialog box opens.

5. In the **Cipher Preference** dialog box, specify which ciphers for the Directory Server to use by selecting them from the list, and click **OK**.

Unless there is a security reason not to use a specific cipher, select all of the ciphers, except for `none`, MD5.

6. In the **Encryption** tab, click **Save**.



CAUTION

Avoid selecting the `none,MD5` cipher because the server will use this option if no other ciphers are available on the client, instead of refusing the connection. The `none,MD5` cipher is not secure because encryption does not occur.

6. Using Certificate-Based Authentication

Directory Server allows certificate-based authentication for the command-line tools (which are LDAP clients) and for replication communications. Certificate-based authentication can occur between:

- An LDAP client connecting to the Directory Server.
- A Directory Server connecting to another Directory Server by replication or chaining.

A single configuration parameter, `nsslapd-certdir`, in `cn=config` in `dse.ldif` lists the directory containing the key, certificate, and security files. The directory name should be unique and specific to the server. For example, the `/etc/dirsrv/slapd-instance_name` directory contains the key and certificate databases only for the Directory Server instance called `instance_name`. That directory will not contain key and certificate databases for any other server or client, nor will any of the key, certificate, or other security-related files for `instance_name` be located in any other directory.



NOTE

The Directory Server 8.0 no longer uses separate files for the key and certificate databases. With the Filesystem Hierarchy Standard, the certificate and key files have been consolidated into a single file, specified in the `nsslapd-certdir` parameter, and the key and certificate file is stored in the `/etc/dirsrv/slapd-instance_name` directory.

Previous versions of Directory Server used a single directory, `/opt/redhat-ds/slapd-instance/alias`, for all security-related files for all servers, and required a unique prefix, such as `slapd-instance-`, for the key, certificate, and security-related files. The Directory Server used the attributes `nsCertFile` and `nsKeyFile` to give the locations for the key and certificate databases.

6.1. Setting up Certificate-Based Authentication

To set up certificate-based authentication, do the following:

1. Create a certificate database for the client and the server or for both servers involved in replication.

In the Directory Server, the certificate database creation automatically takes place when a certificate is installed. For information on creating a certificate database for a client, see [Section 7, “Configuring LDAP Clients to Use SSL”](#).

2. Obtain and install a certificate on both the client and the server or on both servers involved in replication.
3. Enable TLS/SSL on the server or on both servers involved in replication.

For information on enabling TLS/SSL, refer to [Section 4, “Starting the Server with SSL Enabled”](#).



NOTE

If the Red Hat Console connects to Directory Server over TLS/SSL, selecting **Require client authentication** disables communication. This is because, although Red Hat Console supports TLS/SSL, it does not have a certificate to use for client authentication.

4. Map the certificate's distinguished name to a distinguished name known by the directory.

This can set access control for the client when it binds using this certificate.

6.2. Allowing/Requiring Client Authentication

If Red Hat Console is configured to connect to the Directory Server using TLS/SSL *and* the Directory Server *requires* client authentication, the Red Hat Console cannot be used to manage server applications. You must use the appropriate command-line utilities instead.

However, to change the directory configuration to no longer *require* but *allow* client authentication in order to use the Red Hat Console, do the following:

1. Stop the Directory Server. ²

```
service dirsrv stop instance
```

2. Modify the `cn=encryption,cn=config` entry by changing the value of the `nsSSLClientAuth` attribute from `required` to `allowed`.

For information on modifying entries from the command-line, see [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

3. Start the Directory Server.

```
service dirsrv start instance
```

Now start Red Hat Console.

7. Configuring LDAP Clients to Use SSL

For all the users of the Directory Server to use TLS/SSL or certificate-based authentication when they connect using LDAP client applications, they *must* perform the following tasks:

- Create a certificate database.
- Trust the certificate authority (CA) that issues the server certificate.

These operations are sufficient if to ensure that LDAP clients recognize the server's certificate. However, to require the LDAP clients to use their own certificate to authenticate to the directory, make sure that all the directory users obtain and install a personal certificate.



NOTE

Some client applications do not verify that the server has a trusted certificate.

1. On the client system, obtain a client certificate from the CA.
2. Install the client certificate on the client system.

Regardless of how the certificate is sent (either in email or on a web page), there should be a link to click to install the certificate.

Record the certificate information that is sent from the CA, especially the subject DN of the certificate because the server must be configured to map it to an entry in the directory. The client certificate resembles the following:

```
-----BEGIN CERTIFICATE-----
MIICmjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMakGA1UEBh
MCVVMxIzAhBgNVBAoTG1BhbG9va2FWaWxsZSBXaWRnZXRxZLCBjbmuMR0w
GwYDVQQLEExRXaWRnZXQgTWFrZXJzICdSJyBVczEpMCCGA1UEAxMgVGZzdC
```

```
BUZXN0IFRlc3QgVGZzdCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3
WhcNOTgwMzI2MDIzMzU3WjBPMQswCQYDVQGEwJVUzEoMCYGA1UEChMfTm
V0c2NhcGUgRGlyZWNo3
-----END CERTIFICATE-----
```

3. Convert the client certificate into binary format using the `certutil` utility.

```
certutil -L -d certdbPath -n userCertName -r > userCert.bin
```

certdbPath is the directory which contains the certificate database; for example, a user certificate for Mozilla Thunderbird is stored in `$HOME/.thunderbird`. *userCertName* is the name of the certificate, and *userCert.bin* is the name of the output file for binary format.

4. On the server, map the subject DN of the certificate to the appropriate directory entry by editing the `certmap.conf` file.



NOTE

Do not map a certificate-based authentication certificate to a distinguished name under `cn=monitor`. Mapping a certificate to a DN under `cn=monitor` causes the bind operation to fail. Map the certificate to a target located elsewhere in the directory information tree. Make sure that the `verifyCert` parameter is set to `on` in the `certmap.conf` file. If this parameter is not set to `on`, Directory Server simply searches for an entry in the directory that matches the information in the `certmap.conf` file. If the search is successful, it grants access without actually checking the value of the `userCertification` and `userCertificate;binary` attributes.

5. In the Directory Server, modify the directory entry for the user who owns the client certificate to add the `userCertificate` attribute.

- a. Select the **Directory** tab, and navigate to the user entry.
- b. Double-click the user entry, and use the **Property Editor** to add the `userCertificate` attribute, with the `binary` subtype.

When adding this attribute, instead of an editable field, the server provides a **Set Value** button.

- c. Click **Set Value**.

A file selector opens. Use it to select the binary file created in [Section 7, “Configuring LDAP Clients to Use SSL”](#).

For information on using the Directory Server Console to edit entries, refer to [Section 1.3](#),

“Modifying Directory Entries”.

Now TLS/SSL and client authentication can be used with the LDAP clients. For information on how to use TLS/SSL with `ldapmodify`, `ldapdelete`, and `ldapsearch`, see the *Directory Server Configuration, Command, and File Reference*.

Managing SASL

Red Hat Directory Server supports LDAP client authentication through the Simple Authentication and Security Layer (SASL), an alternative to TLS/SSL and a native way for some applications to share information securely.

Directory Server supports SASL authentication using the `DIGEST-MD5` and `GSS-API` mechanisms, allowing Kerberos tickets to authenticate sessions and encrypt data. This chapter describes how to use SASL with Directory Server.

SASL is a framework, meaning it sets up a system that 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 can also set up a security layer for an encrypted session. Directory Server utilizes the `GSS-API` mechanism to encrypt data during sessions.



NOTE

SASL data encryption is not supported for client connections that use TLS/SSL.

1. Authentication Mechanisms

Directory Server support the following SASL encryption mechanisms:

- **EXTERNAL.** The `EXTERNAL` authentication mechanism is utilized by services such as TLS/SSL. It can be used with public keys for strong authentication, such as client certificate-based authentication.
- **CRAM-MD5.** `CRAM-MD5` is a simple challenge-response authentication method that provides no security layer. Red Hat recommends using a more secure mechanism such as `DIGEST-MD5` or `GSS-API`.
- **DIGEST-MD5.** `DIGEST-MD5` is a mandatory authentication method for LDAPv3 servers. While it is not as strong as public key systems or Kerberos authentication methods, it is preferred over plain text passwords and does protect against plain text attacks.
- **Generic Security Services (GSS-API).** Generic Security Services (GSS) is a security API that is the native way for UNIX-based operating systems to access and authenticate Kerberos services. `GSS-API` also supports session encryption, similar to TLS/SSL. (However, `GSS-API` is not compatible with TLS/SSL; they cannot be used simultaneously.) This allows LDAP clients to authenticate with the server using Kerberos version 5 credentials (tickets) and to use network session encryption.



NOTE

GSS-API and, thus, Kerberos are only supported on platforms that have GSS-API support. To use GSS-API, it may be necessary to install the Kerberos client libraries; any required Kerberos libraries will be available through the operating system vendor.

CRAM-MD5, DIGEST-MD5, and GSS-API are *shared secret* mechanisms. The server challenges the client attempting to bind with a *secret*, such as a password, that depends on the mechanism. The user sends back the response required by the mechanism.



NOTE

DIGEST-MD5 requires clear text passwords. The Directory Server requires the clear text password in order to generate the shared secret. Passwords already stored as a hashed value, such as SHA1 *cannot* be used with DIGEST-MD5.

2. 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 preconfigured default maps which handle most common configurations, and customized maps can be created. During a bind attempt, the first matching mapping rule is applied. If only one user identity is returned, the bind is successful; if none or more than one are returned, then the bind fails. Red Hat recommends configuring SASL maps so that only one mapping rule matches the authentication string.



NOTE

SASL proxy authorization is not supported in Directory Server; therefore, Directory Server ignores any SASL `authzid` value supplied by the client.

SASL is configured by entries under a container entry:

```
dn: cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: sasl
```

SASL identity mapping entries are children of this entry:

```
dn: cn=mapping,cn=sasl,cn=config
objectClass: top
objectClass: nsContainer
cn: mapping
```

Mapping entries contain three attributes, `nsSaslMapRegexString`, `nsSaslMapBaseDNTemplate`, and `nsSaslMapFilterTemplate`. The `nsSaslMapping` object class sets these identity mapping parameters.

The `nsSaslMapRegexString` attribute sets variables of the form `\1`, `\2`, `\3`, as in the example, for bind IDs which are filled into the template attributes during a search. For example, this sets up `nsSaslMapping`:

```
dn: cn=mymap,cn=mapping,cn=sasl,cn=config
objectclass:top
objectclass:nsSaslMapping
cn: mymap
nsSaslMapRegexString: \(.*\)\@(.*\)\.\\(.*\)
nsSaslFilterTemplate: (objectclass=inetOrgPerson)
nsSaslBaseDNTemplate: uid=\1,ou=people,dc=\2,dc=\3
```

When a Directory Server receives a SASL bind request with `mconnors@EXAMPLE.COM` as the user ID (authid), the regular expression would fill in the base DN template with `uid=mconnors,ou=people,dc=EXAMPLE,dc=COM` as the user ID, and authentication would proceed 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 broader mapping scheme, such as the following:

```
objectclass: top
objectclass: nsSaslMapping
cn: mymap2
nsSaslMapRegexString: \(.*\)
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This will match any user ID and map to the result of the subtree search with base `ou=People,dc=example,dc=com` and filter `cn=userid`.

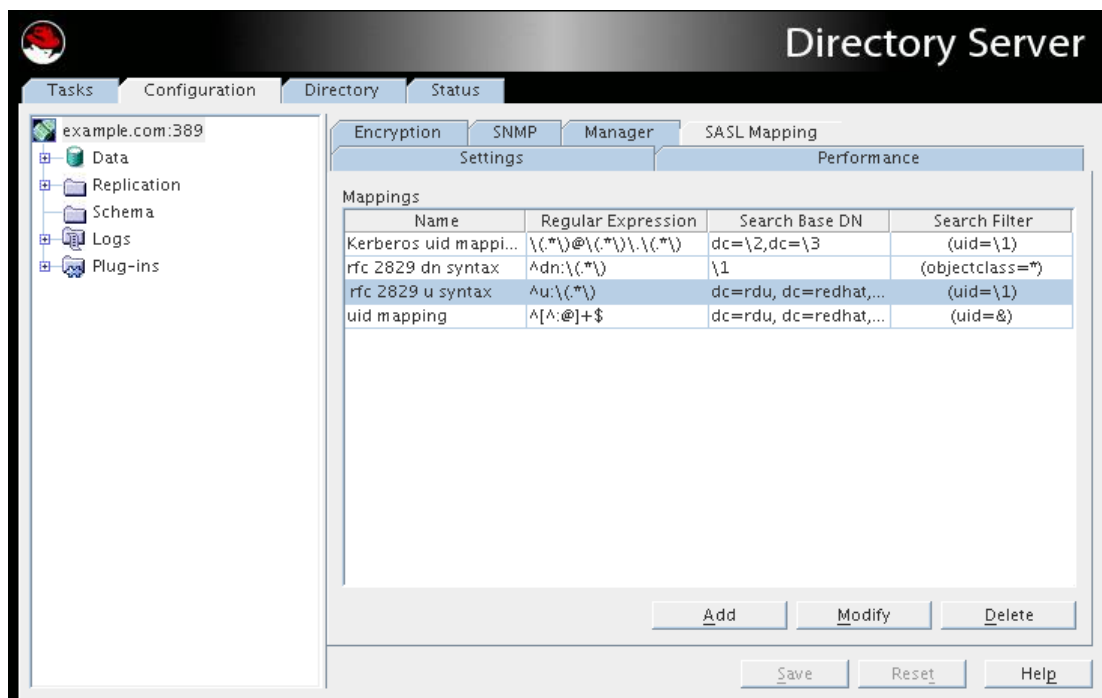
The Directory Server has pre-defined SASL mapping rules to handle some of the most common cases:

- *Kerberos UID Mapping.* This mapping 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 `authid` defines the filter. In this example, the search base would be `dc=example,dc=com` and the filter of `(uid=user)`.
- *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.
- *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.
- *uid Mapping.* This mapping matches an `authid` that is any plain string that does not match the other default mapping rules. It use this value to define a filter of `(uid=value)`. The search base is hard-coded to be the suffix of the default `userRoot` database.

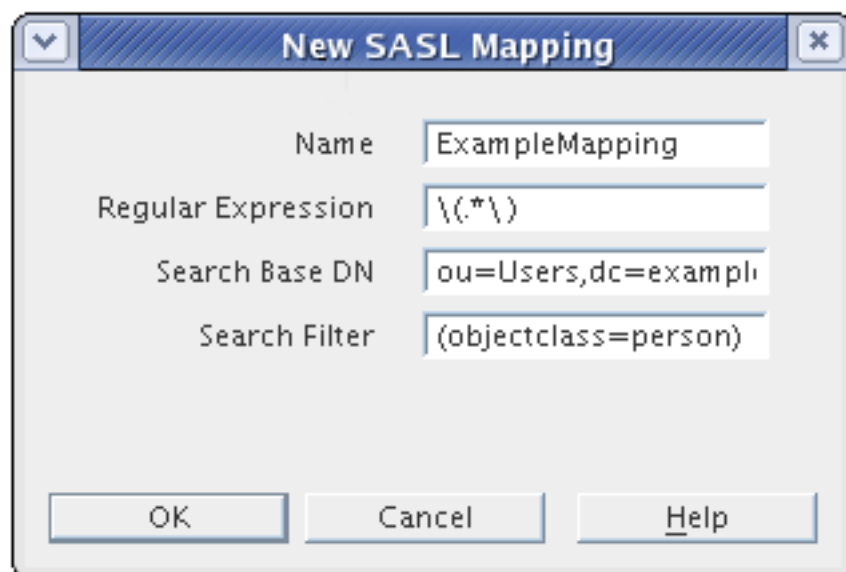
3. Configuring SASL Identity Mapping from the Console

To create a new SASL identity mapping, do the following:

1. In the Directory Server Console, open the **Configuration** tab.
2. Select the **SASL Mapping** tab.



3. To add a new SASL identity mapping, select the **Add** button, and fill in the required values.



- *Name*. This field sets the unique name of the SASL mapping.
- *Regular expression*. This field sets the regular expression used to match the DN components, such as `\\(.*\\)`. This field corresponds to the `nsSaslMapRegexString` value in the SASL mapping LDIF entry.
- *Search base DN*. This field gives the base DN to search to map entries, such as

`ou=People,dc=example,dc=com`. This field corresponds to the `nsSaslMapBaseDNTemplate` value in the SASL mapping LDIF entry.

- *Search filter*. This field gives the search filter for the components to replace, such as `(objectclass=*)`. This field corresponds to the `nsSaslMapFilterTemplate` value in the SASL mapping LDIF entry.

To edit a SASL identity mapping, highlight that identity in the **SASL Mapping** tab, and click **Modify**. Change any values, and save.

To delete a SASL identity mapping, highlight it and hit **Delete**. A dialog box comes up to confirm the deletion.

4. Configuring SASL Identity Mapping from the Command-Line

To configure SASL identity mapping from the command-line, use the `ldapsearch` utility to configure an identity mapping scheme, such as the following:

```
objectclass: top
objectclass: nsSaslMapping
cn: mymap2
nsSaslMapRegexString: \(.*\)
nsSaslMapBaseDNTemplate: ou=People,dc=example,dc=com
nsSaslMapFilterTemplate: (cn=\1)
```

This will match any user ID and map to the result of the subtree search with base `ou=People,dc=example,dc=com` and filter `cn=userId`.

For more information on the `ldapsearch` utility, see [Appendix B, Finding Directory Entries](#).

5. Configuring Kerberos

Kerberos v5 must be deployed on the system to utilize the `GSS-API` mechanism for SASL authentication. [Table 12.1, “Supported Kerberos Systems”](#) summarizes the Kerberos applications supported by various platforms. `GSS-API` and Kerberos client libraries must be installed on the Directory Server host to take advantage of Kerberos services.

Operating System	Kerberos Version
Linux	MIT Kerberos version 5
HP-UX 11i	HP Kerberos version 2.1
Sun Solaris	SEAM 1.0.1

Table 12.1. Supported Kerberos Systems

5.1. Realms

A *realm* is a set of users and the authentication methods for those users to access the realm. A realm resembles a fully-qualified domain name and can be distributed across either a single server or a single domain across multiple machines. A single server instance can also support multiple realms.

Realms are used by the server to associate the DN of the client in the following form, which looks like an LDAP DN:

```
uid=user_name/[server_instance],cn=realm,cn=mechanism,cn=auth
```



NOTE

Kerberos systems treat the Kerberos realm as the default realm; other systems default to the server.

Mike Connors in the `engineering` realm of the European division of `example.com` would have the following association if he tried to access a different server, such as `cyclops`:

```
uid=mconnors/cn=Europe.example.com,  
cn=engineering,cn=gssapi,cn=auth
```

Babara Jensen in the `accounting` realm of `US.example.com` would not have to specify a realm:

```
uid=bjensen,cn=accounting,cn=gssapi,cn=auth
```

If realms are supported by the mechanism and the default realm was not used, *realm* must be specified; otherwise, it is omitted. Currently, only GSS-API supports the concept of realms.

5.2. Configuring the KDC Server

To use GSS-API, the user first obtains a ticket granting ticket (TGT). In many systems, this TGT is issued when the user first logs into the operating system. There are usually command-line utilities provided with the operating system — `kinit`, `klist`, and `kdestroy` — that can be used to acquire, list, and destroy the TGT. The ticket and the ticket's lifetime are parameters in the Kerberos client and server configuration.

Refer to the operating system documentation for information on installing and configuring a Kerberos server (also called a *key distribution center* or KDC). Configuring a KDC for Directory Server is described in [Section 5.3, “Example: Configuring an Example KDC Server”](#).



NOTE

On Red Hat Enterprise Linux, the client-side Kerberos configuration is in the `/etc/krb5.conf`. On Solaris, the client-side Kerberos configuration is in the `/etc/krb5/krb5.conf`.

The HP server and client are separate packages with their own configuration. The server stores config files in `/opt/krb5`. The client is classic MIT and uses `/etc/krb5.conf`. Both the server and client must be configured to have a working Kerberos system.

In order to respond to Kerberos operations, the Directory Server requires access to its own cryptographic key. This key is read by the Kerberos libraries that the server calls, through GSS-API, and the details of how it is found are implementation-dependent. However, in current releases of the supported Kerberos implementations, the mechanism is the same: the key is read from a file called a *keytab* file. This file is created by the Kerberos administrator by exporting the key from the KDC. Either the system default keytab file (typically `/etc/krb5.keytab`) is used, or a service-specific keytab file determined by the value of the `KRB5_KTNAME` environment variable; this environment variable can be set in the `start-slapd` script, which is recommended because it ensures that the variable is properly set each time Directory Server starts.

The Directory Server uses the service name `ldap`. Its Kerberos principal is `ldap/host-fqdn@realm`, like `ldap/dap.corp.example.com/EXAMPLE.COM`. The *host-fqdn* must be the fully-qualified host and domain name, which can be resolved by all LDAP and Kerberos clients through both DNS and reverse DNS lookups. A key with this identity must be stored in the server's *keytab* in order for Kerberos to work.

For information on setting up the service key, see the Kerberos documentation.

5.3. Example: Configuring an Example KDC Server

This example code shows a KDC server configured with the `company.example.com` realm.

```
[libdefaults]
    ticket_lifetime = 24000
    default_realm = COMPANY.EXAMPLE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    ccache_type = 1
    forwardable = true
    proxiable = true
    default_tgs_etypes = des3-hmac-sh1 des-cbc-crc
    default_tkt_etypes = des3-hmac-sh1 des-cbc-crc
    permitted_etypes = des3-hmac-sh1 des-cbc-crc
[realms]
    COMPANY.EXAMPLE.COM = {
        kdc = kdcserver.company.example.com:88
```



```

    admin_server = adminserver.company.example.com:749
    default_domain = company.example.com
}
[appdefaults]
  pam = {
    debug = true
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
    krb4_convert = false
  }
[logging]
  default = FILE:/var/krb5/kdc.log
  kdc = FILE:/var/krb5/kdc.log
  admin_server = FILE:/var/log/kadmind.log

```

5.4. 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 configuration file is in `/etc/sysconfig/dirsrv`.



NOTE

The default configuration file on Red Hat Enterprise Linux and HP-UX is in `/etc/sysconfig`. On Solaris, it is in `/etc/default`.

If there are multiple Directory Server instances and not all of them will use SASL authentication, then there can be instance-specific configuration files created in that directory named `dirsrv-instance`. For example, `dirsrv-example`. The default `dirsrv` file can be used for a single instance.

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/krb5.keytab ; export KRB5_KTNAME

```

For more information on the keytab file, see [Section 5.2, “Configuring the KDC Server”](#).

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 [Chapter 14, Monitoring Directory Server Using SNMP](#).

1. Viewing and Configuring Log Files

Directory Server provides three types of logs to help better manage the directory and tune performance. There are three types of logs:

- Access
- Errors
- Audit

For all types of logs, 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. The following sections describe how to define the log file creation and deletion policy and how to view and configure each type of log.



NOTE

When the server is not running, the log files cannot be viewed in the Directory Server Console, but they can be viewed in the Admin Express. Open the Administration Server URL in a browser:

```
http://hostname:admin_server_port
```

Then log in with the admin login ID and password, and click the link for **Administration Express**.

1.1. Defining a Log File Rotation Policy

For the directory to archive the current log periodically and start a new one, define a log file rotation policy in the Directory Server Console. The log file rotation policy has the following configuration parameters:

- *The access mode or file permissions with which log files are to be created.* The default value is 600. The valid values are any combination of 000 to 777, as they mirror *numbered* or *absolute* UNIX file permissions. This value must be a combination of a 3-digit number, the digits varying from 0 through 7:

- 0 — None
- 1 — Execute only
- 2 — Write only
- 3 — Write and execute
- 4 — Read only
- 5 — Read and execute
- 6 — Read and write
- 7 — Read, write, and execute

In the 3-digit number, the first digit represents the owner's permissions, the second digit represents the group's permissions, and the third digit represents everyone's permissions. When changing the default value, keep in mind that 000 will not allow access to the logs and that allowing write permissions to everyone can result in the logs being overwritten or deleted by anyone.

The newly configured access mode will only affect new logs that are created; the mode will be set when the log rotates to a new file.

- *The maximum number of logs for the directory to keep.* When the directory reaches this number of logs, it deletes the oldest log file in the folder before creating a new log. The default is 10 logs. Do not set this value to 1, or the directory will not rotate the log, and the log will grow indefinitely.
- *The maximum size (in megabytes) for each log file.* To keep from setting a maximum size, type -1 in this field. The default is 100 megabytes. Once a log file reaches this maximum size (or the maximum age), the directory archives the file and starts a new one. Setting the maximum number of logs to 1 causes the directory to ignore this attribute.
- *How often the directory archives the current log file and creates a new one.* The maximum age of the file can be set in minutes, hours, days, weeks, or months. The logs can also be rotated at a particular time of the day; for example, every day at midnight. The default is every day. Setting the maximum number of logs to 1 causes the directory to ignore this attribute.

Each log file includes a title, which identifies the server version, hostname, and port, for ease of archiving or exchanging log files. The title has the following form:

```
Red Hat-Directory/version build_number hostname:port  
( /usr/lib/dirsrv/slapd-instance_name )
```

For example, the first couple of lines of any log files generated by a Directory Server instance may show lines similar to these:

```
Red Hat-Directory/8.0 B2007.188.1157 myhost.example.com:389
(/usr/lib/dirsrv/slapd-example)
```

1.2. Defining a Log File Deletion Policy

For the directory to automatically delete old archived logs, define a log file deletion policy from the Directory Server Console.



NOTE

The log deletion policy only makes sense if there is already a defined log file rotation policy. Log file deletion will not work if there is just one log file. The server evaluates the log file deletion policy at the time of log rotation.

The log file deletion policy can be configured with the following parameters:

- *The maximum size of the combined archived logs.* When the maximum size is reached, the oldest archived log is automatically deleted. To avoid setting a maximum size, type `-1` in this field. The default is 500 megabytes. This parameter is ignored if the maximum number of log files is set to 1.
- *The minimum amount of free disk space.* When the free disk space reaches this minimum value, the oldest archived log is automatically deleted. The default is 5 megabytes. This parameter is ignored if the maximum number of log files is set to 1.
- *The maximum age of log files.* When a log file reaches this maximum age, it is automatically deleted. The default is 1 month. This parameter is ignored if the maximum number of log files is set to 1.

1.3. Access Log

The access log contains detailed information about client connections to the directory.

1.3.1. Viewing the Access Log

To view the access log in the Directory Server Console, do the following:

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Log** folder, and select the **Access Log** icon.

A table displays a list of the last 25 entries in the access log.

- To refresh the current display, click **Refresh**. Select the **Continuous** checkbox for the display to refresh automatically every ten seconds.



NOTE

Continuous log refresh does not work well with log files over 10 megabytes.

- To view an archived access log, select it from the **Select Log** pull-down menu.
- To display a different number of messages, enter the number to view in the **Lines to show** text box, and then click **Refresh**.
- To display messages containing a specified string, enter the string in the **Show only lines containing** text box, and click **Refresh**.

1.3.2. Configuring the Access Log

There are a number of settings that can be configured to customize the access log, including where the directory stores the access log and the creation and deletion policies.

It is also possible to disable access logging for the directory. It may be useful to disable access logging because the access log can grow very quickly; every 2,000 accesses to the directory increases the access log by approximately 1 megabyte. However, before turning off access logging, consider that the access log provides beneficial troubleshooting information.

To configure the access log for the directory, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, expand the **Log** folder, and select the **Access Log** icon.

The access log configuration attributes are displayed in the right pane.

3. To enable access logging, select the **Enable Logging** checkbox.

Clear this checkbox to keep the directory from maintaining an access log. Access logging is enabled by default.

4. Enter the full path and filename for the directory to use for the access log in the **Log File** field. The default path is `/var/log/dirsrv/slaped-instance_name/access`.
5. Set the maximum number of logs, log size, and archive time period.

For information on these parameters, see [Section 1.1, “Defining a Log File Rotation Policy”](#).

6. Set the maximum size of combined archived logs, minimum amount of free disk space, and

maximum age for a log file.

For information on these parameters, see [Section 1.2, “Defining a Log File Deletion Policy”](#).

7. Click **Save**.

The `logconv.pl` Perl script reports the statistical information retrieved from the access log. For more information on `logconv.pl`, refer to the *Directory Server Configuration, Command, and File Reference*.

1.4. Error Log

The error log contains detailed messages of errors and events the directory experiences during normal operations.



WARNING

If the Directory Server fails to write to the errors log, the server sends the message to `syslog` and exits.

1.4.1. Viewing the Error Log

To view the error log, do the following: ¹

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Log** folder, and select the **Error Log** icon.

A table displays a list of the last 25 entries in the error log.

- To refresh the current display, click **Refresh**. Select the **Continuous** checkbox for the display to refresh automatically every ten seconds.



NOTE

Continuous log refresh does not work well with log files over 10 megabytes.

- To view an archived error log, select it from the **Select Log** pull-down menu.
- To specify a different number of messages, enter the number of lines to view in the **Lines to show** text box, and click **Refresh**.

¹ If the Directory Server fails to write to the errors log, the server sends the message to `syslog` and exits.

- To display messages containing a specified string, enter the string in the **Show only lines containing** text box, and click **Refresh**.

1.4.2. Configuring the Error Log

There are several configuration settings for the error log, including where the directory stores the log and what information the directory includes in the log. To configure the error log, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, expand the **Logs** folder, and select the **Error Log** icon.

The error log configuration attributes are displayed in the right pane.

3. Select the **Error Log** tab in the right pane.
4. To enable error logging, select the **Enable Logging** checkbox.

Clear this checkbox to keep the directory from maintaining an error log. Error logging is enabled by default.

5. Enter the full path and filename for the directory to use for the error log in the **Log File** field. The default path is the `/var/log/dirsrv/slapd-instance_name/errors` directory.
6. Set the maximum number of logs, log size, and time period when the file is archived.

For information on these parameters, see [Section 1.1, “Defining a Log File Rotation Policy”](#).

7. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

For information on these parameters, see [Section 1.2, “Defining a Log File Deletion Policy”](#).

8. To set the log level, use the **Ctrl** key and click the options for the directory to include in the **Log Level** list box.

For more information about log level options, see **Log Level** in the *Directory Server Configuration, Command, and File Reference*.



NOTE

Changing these values from the defaults may cause the error log to grow very rapidly, so Red Hat recommends not changing the logging level without being asked to do so by Red Hat technical support.

9. Click **Save**.

1.5. Audit Log

The audit log contains detailed information about changes made to each database as well as to server configuration.

1.5.1. Viewing the Audit Log

Before the audit log can be viewed, audit logging must be enabled for the directory, so the audit log will not be kept. [Section 1.5.2, “Configuring the Audit Log”](#) has more information.

To view the audit log, do the following:

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Log** folder, and select the **Audit Log** icon.

A table displays a list of the last 25 entries in the audit log.

- To refresh the current display, click **Refresh**. Select the **Continuous** checkbox for the display to refresh automatically every ten seconds.



NOTE

Continuous log refresh does not work well with log files over 10 megabytes.

- To view an archived audit log, select it from the **Select Log** pull-down menu.
- To display a different number of messages, enter the number of lines to view in the **Lines to show** text box, and click **Refresh**.
- To display messages containing a specified string, enter the string in the **Show only lines containing** text box, and click **Refresh**.

1.5.2. Configuring the Audit Log

The Directory Server Console can be used to enable and disable audit logging and to specify where the audit log file is stored. To configure audit logging, do the following:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, expand the **Log** folder, and select the **Audit Log** icon.

The audit log configuration attributes are displayed in the right pane.

3. To enable audit logging, select the **Enable Logging** checkbox.

To disable audit logging, clear the checkbox. By default, audit logging is disabled.

4. Enter the full path and filename for the directory to use for the audit log in the field provided.

The default path is `/var/log/dirsrv/slapd-instance_name/audit`.

5. Set the maximum number of logs, log size, and time period when the file is archived.

For information on these parameters, see [Section 1.1, “Defining a Log File Rotation Policy”](#).

6. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

For information on these parameters, see [Section 1.2, “Defining a Log File Deletion Policy”](#).

7. Click **Save**.

2. 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 not automatic log file creation or deletion policies configured. By default, access, error, and audit log files can be found in the following location:

```
/var/log/dirsrv/slapd-instance_name
```

To rotate log files manually, do the following:

1. Shut down the server.²

```
service dirsrv stop instance
```

2. Move or rename the log file being rotated so that the old log file is available for future reference.

3. Restart the server.

```
service dirsrv restart instance
```

3. Monitoring Server Activity

The Directory Server's current activities can be monitored from either the Directory Server

² The command to stop the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, “Starting and Stopping Servers”](#).

Console or the command line. It is also possible to monitor the activity of the caches for all of the database.

3.1. Monitoring the Server from the Directory Server Console

To monitor the server's activities using Directory Server Console, do the following:

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, select **Performance Counters**.

The **Status** tab in the right pane displays current information about server activity. If the server is currently not running, this tab will not provide performance monitoring information.

3. Click **Refresh** to refresh the current display. For the server to continuously update the displayed information, select the **Continuous** checkbox.

The server monitoring information is described in the following tables.

- [Table 13.1, "General Information \(Server\)"](#)
- [Table 13.2, "Resource Summary"](#)
- [Table 13.3, "Current Resource Usage"](#)
- [Table 13.4, "Connection Status"](#)
- [Table 13.5, "Global Database Cache Information"](#)

Field	Description
Server Version	Identifies the current server version.
Configuration DN	Identifies the distinguished name that must be used as a search base to obtain these results using the <code>ldapsearch³</code> command-line utility. This field should read <code>cn=monitor</code> .
Data Version	Provides identification information for the server's data. Usually the information shown here is only relevant if the server supplies replicas to consumer servers. The data version information is supplied as follows: Server hostname. Server port number. Database generation number. <i>Obsolete</i> . A unique identifier that is created only when the directory database is created without a

Field	Description
	machine data entry in the LDIF file. The current changelog number. This is the number corresponding to the last change made to the directory. This number starts at one and increments by one for each change made to the database.
Startup Time on Server	The date and time the server was started.
Current Time on Server	The current date and time on the server.

Table 13.1. General Information (Server)

Resource	Usage Since Startup	Average Per Minute
Connections	The total number of connections to this server since server startup.	Average number of connections per minute since server startup.
Operations Initiated	The total number of operations initiated since server startup. Operations include any client requests for server action, such as searches, adds, and modifies. Often, multiple operations are initiated for each connection.	Average number of operations per minute since server startup.
Operations Completed	The total number of operations completed by the server since server startup.	Average number of operations per minute since server startup.
Entries Sent to Clients	The total number of entries sent to clients since server startup. Entries are sent to clients as the result of search requests.	Average number of entries sent to clients per minute since server startup.
Bytes Sent to Clients	The total number of bytes sent to clients since server startup.	Average number of bytes sent to clients per minute since server startup.

Table 13.2. Resource Summary

Resource	Current Total
Active Threads	The current number of active threads used for handling requests. Additional threads may be

Resource	Current Total
	created by internal server tasks, such as replication or chaining.
Open Connections	The total number of open connections. Each connection can account for multiple operations, and therefore multiple threads.
Remaining Available Connections	The total number of remaining connections that the server can concurrently open. This number is based on the number of currently open connections and the total number of concurrent connections that the server is allowed to open. In most cases, the latter value is determined by the operating system and is expressed as the number of file descriptors available to a task.
Threads Waiting to Write to Client	The total number of threads waiting to write to the client. Threads may not be immediately written when the server must pause while sending data to a client. Reasons for a pause include a slow network, a slow client, or an extremely large amount of information being sent to the client.
Threads Waiting to Read from Client	The total number of threads waiting to read from the client. Threads may not be immediately read if the server starts to receive a request from the client, and then the transmission of that request is halted for some reason. Generally, threads waiting to read are an indication of a slow network or client.
Databases in Use	The total number of databases being serviced by the server.

Table 13.3. Current Resource Usage

Table Header	Description
Time Opened	The time on the server when the connection was initially opened.
Started	The number of operations initiated by this connection.
Completed	The number of operations completed by the server for this connection.
Bound as	The distinguished name used by the client to

Table Header	Description
	bind to the server. If the client has not authenticated to the server, the server displays <code>not bound</code> in this field.
Read/Write	<p>Indicates whether the server is currently blocked for read or write access to the client. There are two possible values:</p> <p>Not blocked means that the server is idle, actively sending data to the client, or actively reading data from the client. Blocked means that the server is trying to send data to the client or read data from the client but cannot. The probable cause is a slow network or client.</p>

Table 13.4. Connection Status

Table Header	Description
Hits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
Tries	The total number of requests performed on the directory since server startup.
Hit Ratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
Pages Read In	The number of pages read from disk into the cache.
Pages Written Out	The number of pages written from the cache back to disk.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from <code>Pages Written Out</code> in that these are discarded read-write pages that have not been modified. Pages discarded

Table Header	Description
	from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts, the better.

Table 13.5. Global Database Cache Information

3.2. Monitoring the Directory Server from the Command Line

The Directory Server's current activities can be monitored using LDAP tools such as `ldapsearch`³, with the following characteristics:

- Search with the attribute filter `objectClass=*`.
- Use the search base `cn=monitor`; the monitoring attributes for the server are found in the `cn=monitor` entry.
- Use the search scope `base`.

For example:

```
ldapsearch -h directory.example.com -p 389 -D "cn=Directory Manager" -w
password -s base
        -b "cn=monitor" "(objectclass=*)"
```

The monitoring attributes for the Directory Server are found in the `cn=monitor` entry. For information on searching the Directory Server, see [Section 2, “Using ldapsearch”](#).

Monitoring the server's activities using `ldapsearch` shows the following information:

Attribute	Description
<code>version</code>	Identifies the directory's current version number.
<code>threads</code>	The current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
<code>connection:fd:opentime:opsinitiated:opscompleted</code>	Provides the following summary information for each open connection (only available if

³ The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

Attribute	Description
	<p>you bind to the directory as Directory Manager):</p> <p><i>fd</i> — The file descriptor used for this connection.</p> <p><i>opentime</i> — The time this connection was opened.</p> <p><i>opsinitiated</i> — The number of operations initiated by this connection.</p> <p><i>opscompleted</i> — The number of operations completed.</p> <p><i>binddn</i> — The distinguished name used by this connection to connect to the directory.</p> <p><i>rw</i> — The field shown if the connection is blocked for read or write.</p> <p>By default, this information is available to Directory Manager. However, the ACI associated with this information can be edited to allow others to access the information.</p>
currentconnections	Identifies the number of connections currently in service by the directory.
totalconnections	Identifies the number of connections handled by the directory since it started.
dtablesiz	Shows the number of file descriptors available to the directory. Each connection requires one file descriptor: one for every open index, one for log file management, and one for <code>ns-slapd</code> itself. Essentially, this value shows how many additional concurrent connections can be serviced by the directory. For more information on file descriptors, refer to the operating system documentation.
readwaiters	Identifies the number of threads waiting to read data from a client.
opsinitiated	Identifies the number of operations the server has initiated since it started.
opscompleted	Identifies the number of operations the server has completed since it started.
entriessent	Identifies the number of entries sent to clients since the server started.
bytessent	Identifies the number of bytes sent to clients since the server started.

Attribute	Description
currenttime	Identifies the time when this snapshot of the server was taken. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
starttime	Identifies the time when the server started. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
nbackends	Identifies the number of back ends (databases) the server services.
backendmonitordn	Identifies the DN of each directory database.

Table 13.6. Server Monitoring Attributes

4. Monitoring Database Activity

The database's current activities can be monitored through Directory Server Console or from the command line.

4.1. Monitoring Database Activity from the Directory Server Console

To monitor the database's activities, do the following:

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Performance Counters** folder, and select the database to monitor.

The tab displays current information about database activity. If the server is currently not running, this tab will not provide performance monitoring information.

3. Click **Refresh** to refresh the currently displayed information. For the directory to continuously update the displayed information, select the **Continuous** checkbox, and then click **Refresh**.

The directory provides database monitoring information as described in the following tables:

- [Table 13.7, “General Information \(Database\)”](#)
- [Table 13.8, “Summary Information”](#)
- [Table 13.9, “Database Cache Information”](#)
- [Table 13.10, “Database File-Specific”](#)

Field	Description
Database	Identifies the type of database being monitored.
Configuration DN	Identifies the distinguished name that must be used as a search base to obtain these results using the <code>ldapsearch</code> command-line utility. ³

Table 13.7. General Information (Database)

Performance Metric	Current Total
Read-Only Status	Shows whether the database is currently in read-only mode. The database is in read-only mode when the <code>nsslapd-readonly</code> attribute is set to <code>on</code> .
Entry Cache Hits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
Entry Cache Tries	The total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against the server since server startup.
Entry Cache Hit Ratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever a search operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the number of entries that the directory maintains in the entry cache by increasing the value of the Maximum Entries in Cache attribute. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.
Current Entry Cache Size (in Bytes)	The total size of directory entries currently present in the entry cache.

Performance Metric	Current Total
Maximum Entry Cache Size (in Bytes)	The size of the entry cache maintained by the directory. This value is managed by the Maximum Cache Size setting. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.
Current Entry Cache Size (in Entries)	The total number of directory entries currently present in the entry cache.
Maximum Entry Cache Size (in Entries)	The maximum number of directory entries that can be maintained in the entry cache. This value is managed by the Maximum Entries in Cache setting. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.

Table 13.8. Summary Information

Performance Metric	Current Total
Hits	The number of times the database cache successfully supplied a requested page. A page is a buffer of the size 2K.
Tries	The number of times the database cache was asked for a page.
Hit Ratio	<p>The ratio of database cache hits to database cache tries. The closer this value is to 100%, the better. Whenever a directory operation attempts to find a portion of the database that is not present in the database cache, the directory has to perform a disk access to obtain the appropriate database page. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory performance drops.</p> <p>To improve this ratio, increase the amount of data that the directory maintains in the database cache by increasing the value of the Maximum Cache Size setting. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.</p>

Performance Metric	Current Total
Pages Read In	The number of pages read from disk into the database cache.
Pages Written Out	The number of pages written from the cache back to disk. A database page is written to disk whenever a read-write page has been modified and then subsequently deleted from the cache. Pages are deleted from the database cache when the cache is full and a directory operation requires a database page that is not currently stored in cache.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from <code>Pages Written Out</code> in that these are discarded read-write pages that have not been modified.

Table 13.9. Database Cache Information

Performance Metric	Current Total
Cache Hits	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
Cache Misses	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
Pages Read In	The number of pages brought to the cache from this file.
Pages Written Out	The number of pages for this file written from cache to disk.

Table 13.10. Database File-Specific

4.2. Monitoring Databases from the Command Line

The directory's database activities can be monitored using any LDAP tool, such as `ldapsearch`³, using the following characteristics:

- Search with the attribute filter `objectClass=*`.
- Use the search base `cn=monitor,cn=database_instance,cn=ldbm database, cn=plugins, cn=config`. *database_instance* is the name of the database to monitor.
- Use the search scope `base`.

For example:

```
ldapsearch -h directory.example.com -s base -p 389 -D "cn=Directory Manager"
-w password
-b "cn=monitor,cn=Example,cn=ldbm database,cn=plugins, cn=config"
"objectclass=*"

```

In this example, the `ldapsearch` operation looks for the `Example` database. For information on searching the directory, see [Section 2, "Using ldapsearch"](#).

Monitoring the server's activities shows the following information:

Attribute	Description
database	Identifies the type of database currently being monitored.
readonly	Indicates whether the database is in read-only mode; 0 means that the server is not in read-only mode, 1 means that it is in read-only mode.
entrycachehits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
entrycachetries	The total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against the server since server startup.
entrycachehitratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever a search operation attempts to find an entry that is not present in the entry cache,

Attribute	Description
	the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the number of entries that the directory maintains in the entry cache by increasing the value of the Maximum Entries in Cache attribute. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.
currententrycachesize	The total size of directory entries currently present in the entry cache.
maxentrycachesize	The maximum number of directory entries that can be maintained in the entry cache. This value is managed by the Maximum Entries in Cache setting. See Section 2, “Tuning Database Performance” for information on changing this value using the Directory Server Console.
dbchehits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
dbcachetries	The total number of requests performed on the directory since server startup.
dbcachehitratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
dbcachepagein	The number of pages read from disk into the cache.
dbcachepageout	The number of pages written from the cache back to disk.
dbcacheroevict	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
dbcacherwevict	The number of read-write pages discarded from the cache to make room for new pages. This value differs from <code>Pages Written Out</code> in that these are discarded read-write pages that

Attribute	Description
	have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
<i>dbfilename-number</i>	The name of the file. <i>number</i> provides a sequential integer identifier (starting at 0) for the file. All associated statistics for the file are given this same numerical identifier.
<i>dbfilecachehit-number</i>	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
<i>dbfilecachemiss-number</i>	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
<i>dbfilepagein-number</i>	The number of pages brought to the cache from this file.
<i>dbfilepageout-number</i>	The number of pages for this file written from cache to disk.

Table 13.11. Directory Server Monitoring Attributes

5. Monitoring Database Link Activity

It is possible to monitor the activity of database links from the command line using the `ldapsearch` command-line utility to return the monitoring attributes that are required. The monitoring attributes are stored in the `cn=monitor,cn=database_link_name,cn=chaining database,cn=plugins,cn=config`.

For example, the `ldapsearch3` command-line utility can be used to retrieve the number of add operations received by a particular database link. For example, this command monitors a database link called `DBLink1`:

```
ldapsearch -h directory.example.com -p 389 -D "cn=Directory Manager" -w
password -s sub -b
      "cn=monitor,cn=DBLink1,cn=chaining database,cn=plugins,cn=config"
      "(objectclass=*)" nsAddCount
```

Table 13.12, "Database Link Monitoring Attributes" lists the database link monitoring attributes

which can be monitored.

Attribute Name	Description
nsAddCount	The number of add operations received.
nsDeleteCount	The number of delete operations received.
nsModifyCount	The number of modify operations received.
nsRenameCount	The number of rename operations received.
nsSearchBaseCount	The number of base-level searches received.
nsSearchOneLevelCount	The number of one-level searches received.
nsSearchSubtreeCount	The number of subtree searches received.
nsAbandonCount	The number of abandon operations received.
nsBindCount	The number of bind request received.
nsUnbindCount	The number of unbinds received.
nsCompareCount	The number of compare operations received.
nsOperationConnectionCount	The number of open connections for normal operations.
nsBindConnectionCount	The number of open connections for bind operations.

Table 13.12. Database Link Monitoring Attributes

For more information about `ldapsearch`, see the *Directory Server Configuration, Command, and File Reference*.

Monitoring Directory Server Using SNMP

The server and database activity monitoring log setup described in [Chapter 13, 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.

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.

2. Configuring the Master Agent

To use the subagent, you must have a master agent that supports AgentX. A common agent is Net-SNMP master agent, which may be available through your operating system vendor or can be downloaded from the Net-SNMP website, <http://www.net-snmp.org>.

The SNMP subagent included with Directory Server uses the AgentX protocol to communicate with the SNMP master agent running on your system. You must make sure that you enable AgentX support on your master agent. For Net-SNMP, add a line containing `agentx master` in the master agent's `snmpd.conf` file. For more details on configuring the master agent for AgentX support, refer to the Net-SNMP website, <http://www.net-snmp.org>.

3. Configuring the Subagent

The Directory Server SNMP subagent is located in installed with the Directory Server tools. On Red Hat Enterprise Linux and Solaris, this is in `/usr/bin/ldap-agent`. On HP-UX, this is in `/opt/dirsrv/bin/ldap-agent`.

3.1. Subagent Configuration File

To use the subagent, first create a subagent configuration file. This file can be named and located wherever you like. This configuration file is used to specify how to communicate with your master agent, logfile location, and which Directory Server instances to monitor.

3.1.1. agentx-master

The `agentx-master` setting tells the subagent how to communicate with the SNMP master agent. If this setting is not specified, the subagent tries to communicate the the master agent through the Unix domain socket `/var/agentx/master`. This is also where the Net-SNMP master agent listens for AgentX communications by default. If you configured your master agent to listen on a different Unix domain socket, you must use the `agentx-master` setting for your subagent to communicate with your master agent by setting the new path for the `agentx-master` parameter. For example:

```
agentx-master /var/snmp/agentx
```

Make sure that the user as whom you are running the subagent has the appropriate permissions to write to this socket.

If the master agent is listening for AgentX communications on a TCP port, the `agentx-master` setting has the hostname and port number. For example:

```
agentx-master localhost:705
```

3.1.2. agent-logdir

The `agent-logdir` setting specifies the directory where the subagent will write its logfile. For example:

```
agent-logdir /var/log
```

If this parameter is not specified, the agent will write its logfile to the same location as your subagent configuration file. The logfile will be named `ldap-agent.log`.

Make sure that the user as whom your subagent is running has write permission to this directory.

3.1.3. server

The `server` setting specifies a Directory Server instance that you want to monitor. You must use one `server` setting for each Directory Server instance. The subagent requires at least one `server` setting to be specified in its configuration file. The `server` setting should be set to the name of the Directory Server instance you would like to monitor. For example:

```
server slapd-phonebook
```

To monitor multiple Directory Server instances, an additional `server` parameter in the subagent configuration file for each instance.

```
server slapd-phonebook
server slapd-example
server slapd-directory
```

3.2. Starting the Subagent

Once your master agent is running and you have created your subagent configuration file, start the subagent. To start your subagent, run the `ldap-agent` program, specifying the absolute path to the subagent configuration file as an argument. For example:

```
ldap-agent /etc/dirsrv/config/ldap-agent.conf
```

To enable extra debug logging, specify the `-D` option during startup:

```
ldap-agent -D /etc/dirsrv/config/ldap-agent.conf
```



NOTE

The Directory Server does not have to be started for the subagent to be started.

To stop your subagent, you must use the kill command against its process ID. Your subagent will print its process ID in its logfile, or you can run `ps -ef | grep ldap-agent` to find the process ID.

3.3. Testing the Subagent

To test your subagent, use any SNMP client tools to query the master agent. Net-SNMP contains simple command-line utilities such as `snmpwalk` and `snmpget`. In order for these tools to use variable names for queries, configure them to load the Directory Server's MIB file. The Directory Server's MIB file, `redhat-ds.mib`, is located in `/usr/share/dirsrv/mibs` on Red Hat Enterprise Linux and Solaris and in `/opt/dirsrv/share/mibs` on HP-UX. There are some additional common required MIB files in this `mibs` directory if you do not already have them with your MIB tools.

The MIB file is not needed for the subagent to operate; it is only required for any SNMP client application to use variable names instead of numeric OIDs to refer to the monitored information provided by the subagent.

Each monitored server instance uses its port number as an index to identify that particular Directory Server instance. For example, querying for the `dsEntityName.389` SNMP variable returns the variable value for a server running on port 389, assuming that instance exists and is being monitored by the subagent.

For details on configuring and using the Net-SNMP command-line tools, check out the Net-SNMP website, <http://www.net-snmp.org>.

4. 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 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.

5. Configuring the Directory Server for SNMP

By default, the Directory Server is ready to be monitored using SNMP as soon as the subagent is configured. However, there are some useful variables in the Directory Server instances which can be configured to help identify the Directory Server instance with SNMP. To configure these SNMP settings from the Directory Server Console, do the following:

1. Select the **Configuration** tab, and then select the topmost entry in the navigation tree in the left pane.
2. Select the **SNMP** tab in the right pane.
3. Enter a description that uniquely describes the directory instance in the **Description** text box.
4. Type the name the company or organization to which the directory belongs in the **Organization** text box.
5. Type the location within the company or organization where the directory resides in the **Location** text box.
6. Type the email address of the person responsible for maintaining the directory in the **Contact** text box.
7. Click **Save**.

6. Using the Management Information Base

The Directory Server's MIB is a file called `redhat-directory.mib`. This MIB contains definitions for variables pertaining to network management for the directory. These variables are known as managed objects. Using the directory MIB and Net-SNMP, you can monitor your

directory like all other managed devices on your network. For more information on using the MIB, refer to [Section 3.3, “Testing the Subagent”](#).

The client tools need to load the Directory Server MIB to use the variable names listed in the following sections.

You can see administrative information about your directory and monitor the server in real-time using the directory MIB. The directory MIB is broken into four distinct tables of managed objects:

- [Section 6.1, “Operations Table”](#)
- [Section 6.2, “Entries Table”](#)
- [Section 6.3, “Entity Table”](#)
- [Section 6.4, “Interaction Table”](#)

6.1. Operations Table

The `Operations Table` provides statistical information about Directory Server access, operations, and errors. [Table 14.1, “Operations Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the `Operations Table` of the `redhat-directory.mib` file.

Managed Object	Description
<code>dsAnonymousBinds</code>	The number of anonymous binds to the directory since server startup.
<code>dsUnauthBinds</code>	The number of unauthenticated binds to the directory since server startup.
<code>dsSimpleAuthBinds</code>	The number of binds to the directory that were established using a simple authentication method (such as password protection) since server startup.
<code>dsStrongAuthBinds</code>	The number of binds to the directory that were established using a strong authentication method (such as SSL or a SASL mechanism like Kerberos) since server startup.
<code>dsBindSecurityErrors</code>	The number of bind requests that have been rejected by the directory due to authentication failures or invalid credentials since server startup.
<code>dsInOps</code>	The number of operations forwarded to this directory from another directory since server startup.
<code>dsReadOps</code>	The number of read operations serviced by

Managed Object	Description
	this directory since application start. The value of this object will always be 0 because LDAP implements read operations indirectly via the search operation.
dsCompareOps	The number of compare operations serviced by this directory since server startup.
dsAddEntryOps	The number of add operations serviced by this directory since server startup.
dsRemoveEntryOps	The number of delete operations serviced by this directory since server startup.
dsModifyEntryOps	The number of modify operations serviced by this directory since server startup.
dsModifyRDNops	The number of modify RDN operations serviced by this directory since server startup.
dsListOps	The number of list operations serviced by this directory since server startup. The value of this object will always be 0 because LDAP implements list operations indirectly via the search operation.
dsSearchOps	The total number of search operations serviced by this directory since server startup.
dsOneLevelSearchOps	The number of one-level search operations serviced by this directory since server startup.
dsWholeSubtreeSearchOps	The number of whole subtree search operations serviced by this directory since server startup.
dsReferrals	The number of referrals returned by this directory in response to client requests since server startup.
dsSecurityErrors	The number of operations forwarded to this directory that did not meet security requirements.
dsErrors	The number of requests that could not be serviced due to errors (other than security or referral errors). Errors include name errors, update errors, attribute errors, and service errors. Partially serviced requests will not be counted as an error.

Table 14.1. Operations Table: Managed Objects and Descriptions

6.2. Entries Table

The `Entries` Table provides information about the contents of the directory entries.

[Table 14.2, “Entries Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the `Entries` Table in the `redhat-directory.mib` file.

Managed Object	Description
<code>dsMasterEntries</code>	The number of directory entries for which this directory contains the master entry. The value of this object will always be 0 (as no updates are currently performed).
<code>dsCopyEntries</code>	The number of directory entries for which this directory contains a copy. The value of this object will always be 0 (as no updates are currently performed).
<code>dsCacheEntries</code>	The number of entries cached in the directory.
<code>dsCacheHits</code>	The number of operations serviced from the locally held cache since application startup.
<code>dsSlaveHits</code>	The number of operations that were serviced from locally held replications (shadow entries). The value of this object will always be 0.

Table 14.2. Entries Table: Managed Objects and Descriptions

6.3. Entity Table

The `Entity` Table contains identifying information about the Directory Server instance. The values for the `Entity` Table are set in the Directory Server Console, as described in [Section 5, “Configuring the Directory Server for SNMP”](#).

[Table 14.3, “Entity Table: Managed Objects and Descriptions”](#) describes the managed objects stored in the `Entity` Table of the `redhat-directory.mib` file.

Managed Object	Description
<code>dsEntityDescr</code>	The description set for the Directory Server instance.
<code>dsEntityVers</code>	The Directory Server version number of the Directory Server instance.
<code>dsEntityOrg</code>	The organization responsible for the Directory Server instance.
<code>dsEntityLocation</code>	The physical location of the Directory Server instance.

Managed Object	Description
dsEntityContact	The name and contact information for the person responsible for the Directory Server instance.
dsEntityName	The name of the Directory Server instance.

Table 14.3. Entity Table: Managed Objects and Descriptions

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 14.4, “*Interaction Table: Managed Objects and Descriptions*” describes the managed objects stored in the `Interaction Table` of the `redhat-directory.mib` file.

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 <code>applIndex</code> , to identify the conceptual row which contains useful information on the (attempted) interaction between the Directory Server (referred to by <code>applIndex</code>) and a peer Directory Server.
dsName	The distinguished name (DN) of the peer Directory Server to which this entry belongs.
dsTimeOfCreation	The value of <code>sysUpTime</code> 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 <code>sysUpTime</code> when the last attempt was made to contact this Directory Server. If the last attempt was made before the network

Managed Object	Description
	management subsystem was initialized, this object will contain a value of zero.
dsTimeOfLastSuccess	The value of <code>sysUpTime</code> 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.

Table 14.4. Interaction Table: Managed Objects and Descriptions

Tuning Directory Server Performance

This chapter describes the tools provided with Red Hat Directory Server to help optimize performance. It also provides tips to improve the performance of the directory.

1. Tuning Server Performance

The server's performance can be managed and improved by limiting the amount of resources the server uses to process client search requests, which is done by defining four settings:

- The maximum number of entries the server returns to the client in response to a search operation (size limit attribute).
- The maximum amount of real time (in seconds) for the server to spend performing a search request (time limit attribute).
- The time (in seconds) during which the server maintains an idle connection before terminating it (idle timeout attribute).
- The maximum number of file descriptors available to the Directory Server (max number of file descriptors attribute).

To configure Directory Server to optimize performance, do the following:

1. In the Directory Server Console, select the **Configuration** tab, and then select the topmost entry in the navigation tree in the left pane.

The tabs that are displayed in the right pane control server-wide configuration attributes.

1. Select the **Performance** tab in the right pane.

The current server performance settings appear.

2. Set the maximum number of entries the server will return to the client in response to a search operation by entering a new value in the **Size Limit** text box.

To keep from setting a limit, type `-1` in this text box.

3. Enter the maximum amount of real time (in seconds) for the server to spend performing a search request in the **Time Limit** text box.

To keep from setting a limit, type `-1` in this text box.

4. Enter the time (in seconds) for the server to maintain an idle connection before terminating it in the **Idle Timeout** text box.

To keep from setting a limit, type zero (0) in this text box.

5. Set the maximum number of file descriptors available to the Directory Server in the **Max Number of File Descriptors** text box. For more information on this parameter, see the *Directory Server Configuration, Command, and File Reference*.

For a better understanding of how these parameters impact the server's search performance, see [Section 1, "About Indexes"](#).

2. Tuning Database Performance

This section is divided into the following parts which describe methods for tuning database performance:

- [Section 2, "Tuning Database Performance"](#)
- [Section 2, "Tuning Database Performance"](#)
- [Section 2, "Tuning Database Performance"](#)
- [Section 2, "Tuning Database Performance"](#)
- [Section 2, "Tuning Database Performance"](#)
- [Section 2, "Tuning Database Performance"](#)

2.1. Optimizing Search Performance

Improve server performance on searches by tuning database settings. The database attributes that affect performance mainly define the amount of memory available to the server. There are two kinds of database caches, one for the default database cache and the other for the entry cache. The server has one default database cache per server, and one entry cache per database.

To improve the cache hit ratio on search operations, increase the amount of data that the Directory Server maintains in the database cache. Do this by increasing the cache size. The maximum values that can be set for these attributes depends on the amount of real memory on the machine. Roughly, the amount of available memory on the machine should always be greater than sum total of the default database cache size and sum of each entry cache size.

Use caution when changing these two attributes. The ability to improve server performance with these attributes depends on the size of the database, the amount of physical memory available on the machine, and whether directory searches are random (that is, if the directory clients are searching for random and widely scattered directory data).

If the database does not fit into memory and if searches are random, attempting to increase the values set on these attributes does not help directory performance. In fact, changing these attributes may harm overall performance.

The following attributes can be tuned:

- The attributes of the database that manages all other database instances. The Directory Server Console only shows the databases that contain the directory data and the `NetscapeRoot` database. However, the server uses another database to manage these. On this database, the following attributes can be changed to improve performance:
 - The amount of memory to make available for all databases (maximum cache size).
 - The maximum number of entries for the server to verify in response to a search request (look-through limit).
 - The amount of memory to make available for import (import cache size).
- The attributes of each database used to store directory data, including the server configuration data in the `NetscapeRoot` database. On these databases, to improve performance, configure the amount of memory to make available for cached entries (memory available for cache).

To configure the default database attributes that apply to all other database instances:

1. In the Directory Server Console, select the **Configuration** tab; then, in the navigation tree, expand the Data Icon, and highlight the Database Settings node.

This displays the Database tabs in the right pane.

1. Select the **LDBM Plug-in Settings** tab in the right pane.

This tab contains the database attributes for all databases stored on this server.

2. In the **Maximum Cache Size** field, enter a value corresponding to the amount of memory to make available for all databases.
3. In the **Look-Through Limit** field, enter the maximum number of entries for the server to check in response to a search request.
4. In the **Import Cache Size** field, enter a value corresponding to the amount of memory in bytes to make available for import. By default, the value is `auto`, and 50% of the free memory is allocated for the import cache. For creating a very large database from LDIF, set this attribute as large as possible, depending on the memory available on the machine. The larger this parameter, the faster the database is created.

To keep from setting a limit, type -1 in this text box. If a user binds to the directory as the Directory Manager, by default the look-through limit is unlimited and overrides any settings specified here.

To configure the attributes of each database that stores the directory data:

5. In the Directory Server Console, select the **Configuration** tab; then, in the navigation tree, expand the Data Icon. Expand the suffix of the database to tune, and highlight the database.

The tabs displayed in the right pane control parameter settings for this database.

1. Select the **Database Settings** tab in the right pane.
2. Enter the amount of memory to make available for cached entries in the **Memory Available for Cache** field.

2.2. Tuning Transaction Logging

Every Directory Server contains a transaction log which writes operations for all the databases it manages. Whenever a directory database operation such as a modify is performed, the server logs the operation to the transaction log. For best performance, the directory does not perform the operation immediately. Instead, the operation is stored in a temporary memory cache on the Directory Server until the operation is completed.

If the server experiences a failure, such as a power outage, and shuts down abnormally, the information about recent directory changes that were stored in the cache is lost. However, when the server restarts, the directory automatically detects the error condition and uses the database transaction log to recover the database.

Although database transaction logging and database recovery are automatic processes that require no intervention, it can be advisable to tune some of the database transaction logging attributes to optimize performance.



Caution

The transaction logging attributes are provided only for system modifications and diagnostics. These settings should be changed only with the guidance of Red Hat Professional Services or Red Hat Technical Support. Setting these attributes and other configuration attributes inconsistently may cause the directory to be unstable.

2.3. Changing the Location of the Database Transaction Log

By default, the database transaction log file is stored in the `/var/lib/dirsrv/slapd-instance_name/db` directory along with the database files themselves. Because the purpose of the transaction log is to aid in the recovery of a directory database that was shut down abnormally, it is a good idea to store the database transaction log on a different disk from the one containing the directory database. Storing the database transaction log on a separate physical disk may also improve directory performance.

To change the location of the database transaction logfile, use the following procedure:

1. Stop the Directory Server¹.

```
service dirsrv stop instance_name
```

2. Use the `ldapmodify`² command-line utility to add the `nsslapd-db-logdirectory` attribute to the `cn=config,cn=ldbm,database,cn=plugins,cn=config` entry. Provide the full path to the log directory in the attribute.

For information on the `nsslapd-db-logdirectory` attribute syntax, see the *Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, see [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

3. Restart Directory Server.

```
service dirsrv start instance_name
```

2.4. Changing the Database Checkpoint Interval

At regular intervals, the Directory Server writes operations logged in the transaction log to the disk and logs a checkpoint entry in the database transaction log. By indicating which changes have already been written to the directory, checkpoint entries indicate where to begin recovery from the transaction log, thus speeding up the recovery process.

By default, the Directory Server is set up to send a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval may increase the performance of directory write operations. However, increasing the checkpoint interval may also increase the amount of time required to recover directory databases after a disorderly shutdown and require more disk space due to large database transaction log files. Therefore, only modify this attribute if you are familiar with database optimization and can fully assess the effect of the change.

¹ To modify the checkpoint interval while the server is running, use the `ldapmodify` command-line utility to add the `nsslapd-db-checkpoint-interval` attribute to the `cn=config,cn=ldbm,database,cn=plugins,cn=config` entry.

² The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozilla` directory on Red Hat Enterprise Linux 5.1386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

the *Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, see [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

2.5. Disabling Durable Transactions

Durable transaction logging means that the temporary database transaction log is, in fact, physically written to disk.

When durable transaction logging is disabled, every directory database operation is written to the database transaction log file but may not be physically written to disk immediately. If a directory change was written to the logical database transaction log file but not physically written to disk at the time of a system crash, the change cannot be recovered. When durable transactions are disabled, the recovered database is consistent but does not reflect the results of any LDAP write operations that completed just before the system crash.

By default, durable database transaction logging is enabled. To disable durable transaction logging, use the following procedure:

1. Stop the Directory Server¹.
2. Use the `ldapmodify`² command-line utility to add the `nsslapd-db-durable-transactions` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry, and set the value of this attribute to `off`.

For information on the syntax of the `nsslapd-db-durable-transactions` attribute, see the *Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, see [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

3. Restart the Directory Server.

2.6. Specifying Transaction Batching

To improve update performance when full transaction durability is not required, use the `nsslapd-db-transaction-batch-val` attribute to specify how many transactions will be batched before being committed to the transaction log. Setting this attribute to a value of greater than 0 causes the server to delay committing transactions until the number of queued transactions is equal to the attribute value. For transaction batching to be valid, the `nsslapd-db-durable-transaction` attribute must be set to `on`.

To specify or modify transaction batching while the server is running, use the `ldapmodify`² command-line utility to add the `nsslapd-db-transaction-batch-val` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

For more information on the syntax and values of the `nsslapd-db-transaction-batch-val` attribute, refer to the *Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, see [Section 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

3. Miscellaneous Tuning Tips

This section covers some common performance-related tips and concepts to remember.

3.1. Avoid Creating Entries Under the `cn=config` Entry in the `dse.ldif` File

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

Although Red Hat recommends that simple user entries not be stored 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.

Administering Directory Server Plug-ins

Plug-ins extend the functionality of the server. Red Hat Directory Server ships with several plug-ins to help manage the directory. This chapter contains general information on the types of plug-ins available and how to enable or disable them.

1. Server Plug-in Functionality Reference

The following tables provide a quick overview of the plug-ins provided with Directory Server, along with their configurable options, configurable arguments, default setting, dependencies, general performance-related information, and further reading. These tables assist in weighing plug-in performance gains and costs and choose the optimal settings for the deployment. The *Further Information* section cross-references further reading, where this is available.

1.1. 7-Bit Check Plug-in

Plug-in Information	Description
Plug-in Name	7-bit check (NS7bitAtt)
Configuration Entry DN	cn=7-bit check,cn=plugins,cn=config
Description	Checks certain attributes are 7-bit clean
Configurable Options	on off
Default Setting	on
Configurable Arguments	List of attributes (<code>uid mail userpassword</code>) followed by "," and then any suffixes for which the check is to occur.
Dependencies	None
Performance Related Information	None
Further Information	If the Directory Server uses non-ASCII characters, such as Japanese characters, turn this plug-in off.

Table 16.1. Details of 7-Bit Check Plug-in

1.2. ACL Plug-in

Plug-in Information	Description
Plug-in Name	ACL Plug-in
Configuration Entry DN	cn=ACL Plugin,cn=plugins,cn=config
Description	ACL access check plug-in

Plug-in Information	Description
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Access control incurs a minimal performance hit. Leave this plug-in enabled since it is the primary means of access control for the server.
Further Information	See Chapter 6, Managing Access Control .

Table 16.2. Details of ACI Plug-in

1.3. ACL Preoperation Plug-in

Plug-in Information	Description
Plug-in Name	ACL Preoperation
Configuration Entry DN	cn=ACL preoperation,cn=plugins,cn=config
Description	ACL access check plug-in
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	database
Performance Related Information	Access control incurs a minimal performance hit. Leave this plug-in enabled since it is the primary means of access control for the server.
Further Information	See Chapter 6, Managing Access Control .

Table 16.3. Details of the ACL Preoperation Plug-in

1.4. Binary Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Binary Syntax
Configuration Entry DN	cn=Binary Syntax,cn=plugins,cn=config
Description	Syntax for handling binary data
Configurable Options	on off

Plug-in Information	Description
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.4. Details of Binary Syntax Plug-in

1.5. Boolean Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Boolean Syntax
Configuration Entry DN	cn=Boolean Syntax,cn=plugins,cn=config
Description	Syntax for handling booleans
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.5. Details of Boolean Syntax Plug-in

1.6. Case Exact String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Case Exact String Syntax
Configuration Entry DN	cn=Case Exact String Syntax,cn=plugins,cn=config
Description	Syntax for handling case-sensitive strings
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None

Plug-in Information	Description
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.6. Details of Case Exact String Syntax Plug-in

1.7. Case Ignore String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Case Ignore String Syntax
Configuration Entry DN	cn=Case Ignore String Syntax,cn=plugins,cn=config
Description	Syntax for handling case-insensitive strings
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.7. Details of Case Ignore String Syntax Plug-in

1.8. Chaining Database Plug-in

Plug-in Information	Description
Plug-in Name	Chaining Database
Configuration Entry DN	cn=Chaining database,cn=plugins,cn=config
Description	Syntax for handling DNs
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	There are many performance related tuning parameters involved with the chaining database. See Section 3, “Creating and Maintaining Database Links” .

Plug-in Information	Description
Further Information	A chaining database is also known as a <i>database link</i> . Database links are described in Section 3, “Creating and Maintaining Database Links” .

Table 16.8. Details of Cloning Database Plug-in

1.9. Class of Service Plug-in

Plug-in Information	Description
Plug-in Name	Class of Service
Configuration Entry DN	cn=Class of Service,cn=plugins,cn=config
Description	Allows for sharing of attributes between entries
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 2, “Assigning Class of Service” .

Table 16.9. Details of Class of Service Plug-in

1.10. Country String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Country String Syntax Plug-in
Configuration Entry DN	cn=Country String Syntax,cn=plugins,cn=config
Description	Syntax for handling countries
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.

Plug-in Information	Description
Further Information	

Table 16.10. Details of Country String Plug-in

1.11. Distinguished Name Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Distinguished Name Syntax
Configuration Entry DN	cn=Distinguished Name Syntax,cn=plugins,cn=config
Description	Syntax for handling DNs
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.11. Details of Distinguished Name Syntax Plug-in

1.12. Generalized Time Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Generalized Time Syntax
Configuration Entry DN	cn=Generalized Time Syntax,cn=plugins,cn=config
Description	Syntax for dealing with dates, times and time zones
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	The Generalized Time String consists of the following:

Plug-in Information	Description
	four digit year two digit month (for example, 01 for January) two digit day, two digit hour two digit minute two digit second decimal part of a second (<i>optional</i>) a time zone indication Red Hat strongly recommends using the Z time zone indication, which stands for Greenwich Mean Time.

Table 16.12. Details of Generalized Time Syntax Plug-in

1.13. Integer Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Integer Syntax
Configuration Entry DN	cn=Integer Syntax,cn=plugins,cn=config
Description	Syntax for handling integers
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.13. Details of Integer Syntax Plug-in

1.14. Internationalization Plug-in

Plug-in Information	Description
Plug-in Name	Internationalization Plug-in
Configuration Entry DN	cn=Internationalization Plugin,cn=plugins,cn=config
Description	Syntax for handling international characters (in DNs)
Configurable Options	on off

Plug-in Information	Description
Default Setting	on
Configurable Arguments	The Internationalization Plug-in has one argument which must not be modified, which specifies the location of the <code>/etc/dirsrv/config/slapd-collations.conf</code> file. This file stores the collation orders and locales used by the Internationalization Plug-in.
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 4, “Searching an Internationalized Directory” and Appendix D, Internationalization .

Table 16.14. Details of Internationalization Plug-in

1.15. Idbm Database Plug-in

Plug-in Information	Description
Plug-in Name	ldbm database Plug-in
Configuration Entry DN	cn=ldbm database plug-in,cn=plugins,cn=config
Description	Implements local databases
Configurable Options	N/A
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	See the <i>Directory Server Configuration, Command, and File Reference</i> for further information on Idbm database plug-in attributes.
Further Information	See Chapter 3, Configuring Directory Databases .

Table 16.15. Details of Idbm Database Plug-in

1.16. Legacy Replication Plug-in

Plug-in Information	Description
Plug-in Name	Legacy Replication Plug-in
Configuration Entry DN	cn=Legacy Replication plug-in,cn=plugins,cn=config
Description	Enables this version of Directory Server to be a consumer of a 4.x supplier
Configurable Options	on off
Default Setting	off
Configurable Arguments	None. This plug-in can be disabled if the server is not (and never will be) a consumer of a 4.x server.
Dependencies	database
Performance Related Information	None
Further Information	See Section 15, “Replication with Earlier Releases” .

Table 16.16. Details of Legacy Replication Plug-in

1.17. Multi-Master Replication Plug-in

Plug-in Information	Description
Plug-in Name	Multi-master Replication Plug-in
Configuration Entry DN	cn=Multimaster Replication plugin,cn=plugins,cn=config
Description	Enables replication between two Directory Servers
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	database
Performance Related Information	N/A
Further Information	This plug-in can only be turned off if there is only one server, which will never replicate. See also Chapter 8, Managing Replication .

Table 16.17. Details of Multi-Master Replication Plug-in

1.18. Octet String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Octet String Syntax
Configuration Entry DN	cn=Octet String Syntax,cn=plugins,cn=config
Description	Syntax for handling octet strings
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.18. Details of Octet String Syntax Plug-in

1.19. CLEAR Password Storage Plug-in

Plug-in Information	Description
Plug-in Name	CLEAR
Configuration Entry DN	cn=CLEAR,cn=Password Storage Schemes,cn=plugins, cn=config
Description	CLEAR password storage scheme used for password encryption
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 1, “Managing the Password Policy” .

Table 16.19. Details of CLEAR Password Storage Plug-in

1.20. CRYPT Password Storage Plug-in

Plug-in Information	Description
Plug-in Name	CRYPT
Configuration Entry DN	cn=CRYPT,cn=Password Storage

Plug-in Information	Description
	Schemes,cn=plugins, cn=config
Description	CRYPT password storage scheme used for password encryption
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 1, “Managing the Password Policy” .

Table 16.20. Details of CRYPT Password Storage Plug-in

1.21. NS-MTA-MD5 Password Storage Plug-in

Plug-in Information	Description
Plug-in Name	NS-MTA-MD5
Configuration Entry DN	cn=NS-MTA-MD5,cn=Password Storage Schemes,cn=plugins, cn=config
Description	NS-MTA-MD5 password storage scheme for password encryption
Configurable Options	on off
Default Setting	off
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Red Hat recommends leaving this plug-in running at all times.
Further Information	Passwords cannot be encrypted using the NS-MTA-MD5 password storage scheme. The storage scheme is present in Directory Server only for reasons of backward compatibility. See Section 1, “Managing the Password Policy” .

Table 16.21. Details of NS-MTA-MD5 Password Storage Plug-in

1.22. SHA Password Storage Plug-in

Plug-in Information	Description
Plug-in Name	SHA
Configuration Entry DN	cn=SHA, cn=Password Storage Schemes, cn=plugins, cn=config cn=SHA256, cn=Password Storage Schemes, cn=plugins, cn=config cn=SHA384, cn=Password Storage Schemes, cn=plugins, cn=config cn=SHA512, cn=Password Storage Schemes, cn=plugins, cn=config
Description	SHA password storage scheme for password encryption
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	If the directory does not contain passwords encrypted using the SHA password storage scheme, this plug-in can be turned off. SHA is only included for compatibility with earlier releases; Red Hat recommends use SSHA rather than SHA because SSHA is a far more secure option.
Further Information	See Section 1, “Managing the Password Policy” .

Table 16.22. Details of SHA Password Storage Plug-in

1.23. SSHA Password Storage Plug-in

Plug-in Information	Description
Plug-in Name	SSHA
Configuration Entry DN	cn=SSHA, cn=Password Storage Schemes, cn=plugins, cn=config cn=SSHA256, cn=Password Storage Schemes, cn=plugins, cn=config cn=SSHA384, cn=Password Storage Schemes, cn=plugins, cn=config cn=SSHA512, cn=Password Storage

Plug-in Information	Description
	Schemes,cn=plugins,cn=config
Description	SSHA password storage scheme for password encryption
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 1, “Managing the Password Policy” .

Table 16.23. Details of SSHA Password Storage Plug-in

1.24. Postal Address String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Postal Address Syntax
Configuration Entry DN	cn=Postal Address Syntax,cn=plugins,cn=config
Description	Syntax used for handling postal addresses
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.24. Details of Postal Address String Syntax Plug-in

1.25. PTA Plug-in

Plug-in Information	Description
Plug-in Name	Pass-Through Authentication Plug-in
Configuration Entry DN	cn=Pass Through Authentication,cn=plugins,cn=config

Plug-in Information	Description
Description	Enables pass-through authentication, the mechanism which allows one directory to consult another to authenticate bind requests. This plug-in is not listed in the Directory Server Console if the same server is used for the user directory and configuration directory.
Configurable Options	on off
Default Setting	off
Configurable Arguments	ldap ldaps://authDS/subtree
Dependencies	None
Performance Related Information	Pass-through authentication slows down bind requests a little because they have to make an extra hop to the remote server. See Chapter 17, Using the Pass-through Authentication Plug-in .
Further Information	See Chapter 17, Using the Pass-through Authentication Plug-in .

Table 16.25. Details of PTA Plug-in

1.26. Referential Integrity Postoperation Plug-in

Plug-in Information	Description
Plug-in Name	Referential Integrity Post-Operation
Configuration Entry DN	cn=Referential Integrity Post operation,cn=plugins, cn=config
Description	Enables the server to ensure referential integrity
Configurable Options	All configuration and on off
Default Setting	off
Configurable Arguments	<p>When enabled, the post-operation Referential Integrity Plug-in performs integrity updates on the <i>member</i>, <i>uniquemember</i>, <i>owner</i> and <i>seeAlso</i> attributes immediately after a delete or rename operation. The plug-in can be reconfigured to perform integrity checks on all other attributes:</p> <ul style="list-style-type: none"> • Check for referential integrity.

Plug-in Information	Description
	<p>-1= no check for referential integrity 0= check for referential integrity is performed immediately Positive integer= request for referential integrity is queued and processed at a later stage. This positive integer serves as a wake-up call for the thread to process the request at intervals corresponding to the integer (number of seconds) specified.</p> <ul style="list-style-type: none"> Log file for storing the change; for example <code>/var/log/dirsrv/slapd-instance_name/referint.</code> All the additional attribute names to be checked for referential integrity.
Dependencies	Database
Performance Related Information	The Referential Integrity Plug-in should be enabled only on one master in a multimaster replication environment to avoid conflict resolution loops. When enabling the plug-in on chained servers, be sure to analyze the performance resource and time needs as well as integrity needs. All attributes specified must be indexed for both presence and equality.
Further Information	See Chapter 10, Managing Indexes for information about how to index attributes used for referential integrity checking.

Table 16.26. Details of Referential Integrity Post-Operation Plug-in

1.27. Retro Changelog Plug-in

Plug-in Information	Description
Plug-in Name	Retro Changelog Plug-in
Configuration Entry DN	<code>cn=Retro Changelog Plug-in,cn=plugins,cn=config</code>
Description	Used by LDAP clients for maintaining application compatibility with Directory Server 4.x versions. Maintains a log of all changes occurring in the Directory Server. The retro

Plug-in Information	Description
	changelog offers the same functionality as the changelog in the 4.x versions of Directory Server. This plug-in exposes the <code>cn=changelog</code> suffix to clients, so that clients can use this suffix with or without persistent search for simple sync applications.
Configurable Options	on off
Default Setting	off
Configurable Arguments	See the <i>Directory Server Configuration, Command, and File Reference</i> for further information on the two configuration attributes for the Retro Changelog Plug-in.
Dependencies	None
Performance Related Information	May slow down Directory Server update performance.
Further Information	See Chapter 8, Managing Replication .

Table 16.27. Details of Retro Changelog Plug-in

1.28. Roles Plug-in

Plug-in Information	Description
Plug-in Name	Roles Plug-in
Configuration Entry DN	<code>cn=Roles Plugin,cn=plugins,cn=config</code>
Description	Enables the use of roles in the Directory Server.
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	See Section 1, "Using Roles" .

Table 16.28. Details of Roles Plug-in

1.29. Space Insensitive String Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Space Insensitive String Syntax
Configuration Entry DN	cn=Space Insensitive String Syntax,cn=plugins,cn=config
Description	Syntax for handling space-insensitive values.
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	This plug-in enables the Directory Server to support <i>space and case insensitive</i> values. Applications can search the directory using entries with ASCII space characters. For example, a search or compare operation that uses <code>John Doe</code> will match entries that contain <code>johndoe</code> , <code>john doe</code> , and <code>John Doe</code> if the attribute's schema has been configured to use the space insensitive syntax. For more information about finding directory entries, see Appendix B, Finding Directory Entries .

Table 16.29. Details of Space Insensitive String Syntax Plug-in

1.30. State Change Plug-in

Plug-in Information	Description
Plug-in Name	State Change Plug-in
Configuration Entry DN	cn=State Change Plugin,cn=plugins,cn=config
Description	Enables state-change-notification service.
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	
Further Information	

Table 16.30. Details of State Change Plug-in

1.31. Telephone Syntax Plug-in

Plug-in Information	Description
Plug-in Name	Telephone Syntax
Configuration Entry DN	cn=Telephone Syntax,cn=plugins,cn=config
Description	Syntax for handling telephone numbers
Configurable Options	on off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.31. Details of Telephone Syntax Plug-in

1.32. UID Uniqueness Plug-in

Plug-in Information	Description
Plug-in Name	UID Uniqueness Plug-in
Configuration Entry DN	cn=UID Uniqueness,cn=plugins,cn=config
Description	Checks that the values of specified attributes are unique each time a modification occurs on an entry. For example, most sites require that a user ID and email address be unique.
Configurable Options	on off
Default Setting	off
Configurable Arguments	To check for UID attribute uniqueness in all listed subtrees, enter <code>uid "DN" "DN" . . .</code> . However, to check for UID attribute uniqueness when adding or updating entries with the <code>requiredObjectClass</code> , enter <code>attribute="uid" MarkerObjectclass = "ObjectClassName" and, optionally requiredObjectClass = "ObjectClassName"</code> . This starts checking for the required object classes from the parent

Plug-in Information	Description
	entry containing the <i>ObjectClass</i> as defined by the <i>MarkerObjectClass</i> attribute.
Dependencies	N/A
Performance Related Information	<p>This plug-in may slow down Directory Server performance. In a multi-master replication environment, the UID Uniqueness Plug-in will not work at all and should therefore not be enabled.</p> <p>Additionally, this plug-in does not work with referrals the UID Uniqueness Plug-in fails with an operations error if it receives any other error than <code>noSuchObject</code> (meaning that the entry does not already exist), which prevents the new entry from being added. The referral on the subtree returns a different error message, so trying to add a new entry to a subtree with a referral while the UID Uniqueness Plug-in is enabled will fail. To prevent being blocked by such an operations error, disable the plug-in on the server where the referral is created. To run a UID uniqueness check, make sure that the plug-in is only active on the last of the referred-to servers to prevent it from blocking the referral mechanism.</p>
Further Information	See Chapter 18, Using the Attribute Uniqueness Plug-in .

Table 16.32. Details of UID Uniqueness Plug-in

1.33. URI Plug-in

Plug-in Information	Description
Plug-in Name	URI Syntax
Configuration Entry DN	cn=URI Syntax,cn=plugins,cn=config
Description	Syntax for handling URIs (Unique Resource Identifiers), including URLs (Unique Resource Locators)
Configurable Options	on off
Default Setting	on

Plug-in Information	Description
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. Leave this plug-in running at all times.
Further Information	

Table 16.33. Details of URI Plug-in

2. Enabling and Disabling Plug-ins

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

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

```
service dirsrv restart instance_name
```

To disable or enable a plug-in through the command line, use the `ldapmodify` utility to edit the value of the `nsslapd-pluginEnabled` attribute. For example: ²

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

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

¹ The command to restart the Directory Server on platforms other than Red Hat Enterprise Linux is described in [Section 3, “Starting and Stopping Servers”](#).

² The LDAP tools referenced in this guide are Mozilla LDAP, installed with Directory Server in the `/usr/lib/mozldap` directory on Red Hat Enterprise Linux 5 i386; directories for other platforms are listed in [Section 2, “LDAP Tool Locations”](#). However, Red Hat Enterprise Linux systems also include LDAP tools from OpenLDAP. It is possible to use the OpenLDAP commands as shown in the examples, but you must use the `-x` argument to disable SASL and allow simple authentication.

Using the Pass-through Authentication Plug-in

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.

1. How Directory Server Uses PTA

If the configuration directory and the user directory are installed on separate instances of Directory Server, the setup program automatically sets up PTA to allow the Configuration Administrator user (usually `admin`) to perform administrative duties.

PTA is required in this case because the `admin` user entry is stored under `o=NetscapeRoot` suffix in the configuration directory. Therefore, attempts to bind to the user directory as `admin` would normally fail. PTA allows the user directory to transmit the credentials to the configuration directory, which verifies them. The user directory then allows the `admin` user to bind.

The user directory in this example acts as the *PTA Directory Server*, the server that passes through bind requests to another Directory Server. The configuration directory acts as the *authenticating directory*, the server that contains the entry and verifies the bind credentials of the requesting client.

The *pass-through subtree* is the subtree *not* present on the PTA directory. When a user's bind DN contains this subtree, the user's credentials are passed on to the authenticating directory.



NOTE

The PTA Plug-in may not be listed in the Directory Server Console the same server instance is used for the user directory and the configuration directory.

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 configuration database and suffix, `o=NetscapeRoot`. In this example, the server name is `configdir.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. When the user directory is set up on machine B, the setup script prompts for the LDAP URL of the configuration directory on machine A.
4. The setup program enables the PTA Plug-in and configures it to use the configuration directory LDAP URL.

This entry contains the LDAP URL for the configuration directory. For example:

```
dn: cn=Pass Through Authentication,cn=plugins,  
...  
nsslapd-pluginEnabled: on  
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot  
...
```

The user directory is now configured to send all bind requests for entries with a DN containing `o=NetscapeRoot` to the configuration directory `configdir.example.com`.

5. When installation is complete, the `admin` user attempts to connect to the user directory to begin adding users.
6. The setup program adds the `admin` user's entry to the directory as `uid=admin, ou=TopologyManagement, o=NetscapeRoot`. So the user directory passes the bind request through to the configuration directory as defined by the PTA Plug-in configuration.
7. The configuration directory authenticates the user's credentials and sends the information back to the user directory.
8. The user directory allows the `admin` user to bind.

2. 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. There are only two attributes in this entry that are significant:

- `nsslapd-pluginEnabled`, which sets whether the plug-in is enabled or disabled. The value for this attribute can be `on` or `off`.
- `nsslapd-pluginarg0`, which points to the configuration directory. The value for this attribute is the LDAP URL of the server and suffix to which to pass the bind requests, along with the optional parameters, `maxconns`, `maxops`, `timeout`, `ldver`, `connlifetime`.

The variable components of the PTA plug-in syntax are described in [Table 17.1, "PTA Plug-in Parameters"](#).



NOTE

The LDAP URL (`ldap|ldaps://authDS/subtree`) must be separated from the optional parameters (*maxconns*, *maxops*, *timeout*, *ldver*, *connlifetime*) by a single space. If 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 4.2](#), “*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.

Variable	Definition
state	Defines whether the plug-in is enabled or disabled. Acceptable values are <code>on</code> or <code>off</code> . See Section 3.1 , “ <i>Turning the Plug-in On or Off</i> ” for more information.
ldap ldaps	Defines whether SSL is used for communication between the two Directory Servers. See Section 3.2 , “ <i>Configuring the Servers to Use a Secure Connection</i> ” for more information.
authDS	The authenticating directory hostname. The port number of the Directory Server can be given by adding a colon and then the port number. For example, <code>ldap://dirserver.example.com:389/</code> . If the port number is not specified, the PTA server attempts to connect using either of the standard ports: Port 389 if <code>ldap://</code> is specified in the URL. Port 636 if <code>ldaps://</code> is specified in the URL. See Section 3.3 , “ <i>Specifying the Authenticating Directory Server</i> ” for more information.

Variable	Definition
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 Section 3.4, “Specifying the Pass-through Subtree” for more information. This subtree must not exist on this server. To pass the bind requests for <code>o=NetscapeRoot</code> to the configuration directory, the subtree <code>o=NetscapeRoot</code> 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 3. See Section 3.5, “Configuring the Optional Parameters” 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 5. See Section 3.5, “Configuring the Optional Parameters” 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 300 seconds (five minutes). Specify zero (0) to indicate no time limit should be enforced. See Section 3.5, “Configuring the Optional Parameters” 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 Section 3.5, “Configuring the Optional Parameters” for more information.
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

Variable	Definition
	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 300 seconds (five minutes). See Section 3.5, “Configuring the Optional Parameters” for more information.

Table 17.1. PTA Plug-in Parameters

3. Configuring the PTA Plug-in

The only method for configuring the PTA plug-in is to modify the entry `cn=Pass Through Authentication,cn=plugins,cn=config`. To modify the PTA configuration, do the following:

1. Use the `ldapmodify` command to modify `cn=Pass Through Authentication,cn=plugins,cn=config`.
2. Restart Directory Server.¹

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 2, “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 3.1, “Turning the Plug-in On or Off”](#)

¹ The commands to start and stop the Directory Server on platforms other than Red Hat Enterprise Linux are described in [Section 3, “Starting and Stopping Servers”](#).

- [Section 3.3, “Specifying the Authenticating Directory Server”](#)
- [Section 3.4, “Specifying the Pass-through Subtree”](#)
- [Section 3.5, “Configuring the Optional Parameters”](#)

3.1. Turning the Plug-in On or Off

To turn the PTA Plug-in on from the command line, do the following:

1. Use the `ldapmodify` command to update the plug-in configuration:

```
ldapmodify -p 389 -D "cn=Directory Manager" -w password -h example

dn: cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

2. Restart the server. ¹

```
service dirsrv restart instance_name
```

To disable the plug-in, change the `nsslapd-pluginenabled` attribute value from `on` to `off`. Whenever the PTA Plug-in is enabled or disabled from the command line, the server must be restarted.

3.2. Configuring the Servers to Use a Secure Connection

The PTA directory can be configured to communicate with the authenticating directory over SSL by specifying LDAPS in the LDAP URL of the PTA directory. For example:

```
nsslapd-pluginarg0: ldaps://ldap.example.com:636/o=NetscapeRoot
```

3.3. 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 defines 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 hostname, as described in [Table 17.1, “PTA Plug-in Parameters”](#).

1. Use `ldapmodify` edit the PTA Plug-in entry.

```
ldapmodify -p 389 -D "cn=Directory Manager" -w password -h example

dn: cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot
```

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. ¹

```
service dirsrv restart instance_name
```

3.4. 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 `ldapmodify` command to import the LDIF file into the directory.

```
ldapmodify -p 389 -D "cn=Directory Manager" -w password -h example

dn: cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot
```

For information on the variable components in this syntax, see [Table 17.1, “PTA Plug-in Parameters”](#).

2. Restart the server. ¹

```
service dirsrv restart instance_name
```

3.5. Configuring the Optional Parameters

Additional parameters that control the PTA connection can be set with the LDAP URL.

```
ldap|ldaps://authDS/subtree maxconns, maxops, timeout, ldver, connlifetime
```

- 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 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*.

1. Use `ldapmodify` to edit the plug-in entry.

```
ldapmodify -p 389 -D "cn=Directory Manager" -w password -h example  
  
dn: cn=Pass Through Authentication,cn=plugins,cn=config  
changetype: modify  
replace: nsslapd-pluginarg0
```

```
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot
3,5,300,3,300
```

(In this example, each of the optional parameters is set to its default value.) 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. ¹

```
service dirsrv restart instance_name
```

4. PTA Plug-in Syntax Examples

This section contains the following examples of PTA Plug-in syntax in the `dse.ldif` file:

- [Section 4.1, “Specifying One Authenticating Directory Server and One Subtree”](#)
- [Section 4.2, “Specifying Multiple Authenticating Directory Servers”](#)
- [Section 4.3, “Specifying One Authenticating Directory Server and Multiple Subtrees”](#)
- [Section 4.4, “Using Non-Default Parameter Values”](#)
- [Section 4.5, “Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers”](#)

4.1. 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 hostname 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
...
```

4.2. 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-separate 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*.

4.3. 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
...
```

4.4. 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
...
```

4.5. 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 7,7,300,3,300
nsslapd-pluginarg1:ldap://config2dir.example.com/dc=example,dc=com
7,7,300,3,300
...
```


Using the Attribute Uniqueness Plug-in

The Attribute Uniqueness Plug-in can be used to ensure that the new or edited attributes always have unique values in the directory. A new instance of the Attribute Uniqueness Plug-in must be created for every attribute for which values must be unique. The Attribute Uniqueness Plug-in can enforce the uniqueness of the value for any attribute.

1. Overview of the Attribute Uniqueness Plug-in

The Attribute Uniqueness Plug-in is a preoperation plug-in. This means that the plug-in checks all update operations *before* the server performs an LDAP operation. The plug-in determines whether the operation applies to an attribute and a suffix that it is configured to monitor.

If an update operation applies to an attribute and suffix monitored by the plug-in and it would cause two entries to have the same attribute value, then the server terminates the operation and returns an `LDAP_CONSTRAINT_VIOLATION` error to the client.

Each instance of the Attribute Uniqueness Plug-in performs a check on a single attribute for one or more subtrees. To check uniqueness of several attributes, a separate instance of the plug-in must be created for each attribute to check.

The Attribute Uniqueness Plug-in can operate in specific, user-defined ways:

- It can check every entry in the specified subtrees.

For example, if a company, `example.com`, hosts the directories for `example_a.com` and `example_b.com`, when an entry such as `uid=jdoe,ou=people,o=example_a,dc=example,dc=com` is added, uniqueness needs to be enforced only in the `o=example_a,dc=example,dc=com` subtree. This is done by listing the DN of the subtree explicitly in the Attribute Uniqueness Plug-in configuration.

This configuration option is explained in more detail in [Section 4.3.2, “Specifying a Suffix or Subtree”](#).

- Specify an object class pertaining to an entry in the DN of the updated entry and perform the uniqueness check on all the entries beneath it.

This option is useful in hosted environments. For example, when adding an entry such as `uid=jdoe,ou=people,o=example_a,dc=example,dc=com`, enforce uniqueness under the `o=example_a,dc=example,dc=com` subtree without listing this subtree explicitly in the configuration but, instead, by indicating a *marker object class*. If the marker object class is set to `organization`, the uniqueness check algorithm locates the entry in the DN that has this object class (`o=example_a`) and performs the check on all entries beneath it.

Additionally, it is possible to check uniqueness only if the updated entry includes a specified object class. For example, a check may be performed only if the updated entry includes `objectclass=inetorgperson`.

This configuration option is explained in more detail in [Section 4.3.3, “Using the `markerObjectClass` and `requiredObjectClass` Keywords”](#).

For information on using the Attribute Uniqueness Plug-in in a replicated environment, see [Section 6, “Replication and the Attribute Uniqueness Plug-in”](#).

Directory Server provides a default instance of the Attribute Uniqueness Plug-in, the UID Uniqueness Plug-in, to ensure that values given to the `uid` attribute are unique in the root suffix (the suffix corresponding to the `userRoot` database) configured when the Directory Server was first set up.

This plug-in is disabled by default because it affects the operation of multi-master replication. For information on using the attribute uniqueness plug-in in a replicated environment, refer to [Section 6, “Replication and the Attribute Uniqueness Plug-in”](#).

2. Attribute Uniqueness Plug-in Syntax

Configuration information for the Attribute Uniqueness Plug-in is specified in an entry under `cn=plugins,cn=config` entry. There are two possible syntaxes for `nsslapd-pluginarg` attributes.



NOTE

To enforce uniqueness of another attribute than the ones in these example, copy and paste the default Attribute Uniqueness Plug-in entry, and being care to change only the attributes described here.

Use the following syntax to perform the uniqueness check under a suffix or subtree:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
...
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute_name
nsslapd-pluginarg1: dn1
nsslapd-pluginarg2: dn2
...
```

- Any value can be given to the `cn` attribute to name the plug-in. The name should be descriptive.

- The *cn* attribute does not contain the name of the attribute which is checked for uniqueness.
- Only one attribute can be specified on which the uniqueness check will be performed.
- It is possible to specify several DNs of suffixes or subtrees in which to perform the uniqueness check by incrementing the *nsslapd-pluginarg* attribute suffix by one each time.

The variable components of the Attribute Uniqueness Plug-in syntax are described in [Table 18.1, “Attribute Uniqueness Plug-in Variables”](#).

Use the following syntax to specify to perform the uniqueness check below an entry containing a specified object class:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
...
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute=attribute_name
nsslapd-pluginarg1: markerObjectClass=objectclass1
nsslapd-pluginarg2: requiredObjectClass=objectclass2
...
```

- Any value can be given to the *cn* attribute to name the plug-in. The name should be descriptive.
- The *cn* attribute does not contain the name of the attribute which is checked for uniqueness.
- Only one attribute can be specified on which the uniqueness check will be performed.
- If the *nsslapd-pluginarg0* attribute begins with *attribute=attribute_name*, then the server expects the *nsslapd-pluginarg1* attribute to include a *markerObjectClass* value.

The variable components of the attribute uniqueness plug-in syntax are described in [Table 18.1, “Attribute Uniqueness Plug-in Variables”](#).

Variable	Definition
<i>descriptive_plugin_name</i>	Specifies the name of this instance of the Attribute Uniqueness Plug-in. It is not required that the name of the attribute for which to ensure uniqueness be included, but it is advisable. For example, <i>cn=mail uniqueness,cn=plugins,cn=config</i> .
<i>state</i>	Defines whether the plug-in is enabled or disabled. Acceptable values are <i>on</i> or <i>off</i> . See Section 4.3.1, “Turning the Plug-in On or Off” for more information.
<i>attribute_name</i>	The name of the attribute for which to ensure unique values. Only one attribute can be

Variable	Definition
	named.
<i>dn</i>	The DN of the suffix or subtree in which to ensure attribute uniqueness. To specify several suffixes or subtrees, increment the suffix of the <i>nsslapd-pluginarg</i> attribute by one for each additional suffix or subtree.
attribute= <i>attribute_name</i>	The name of the attribute for which to ensure unique values. Only one attribute can be named.
markerObjectClass= <i>objectclass1</i>	Attribute uniqueness will be checked under the entry belonging to the DN of the updated entry that has the object class specified in the <i>markerObjectClass</i> keyword. Do not include a space before or after the equals sign.
requiredObjectClass= <i>objectclass2</i>	<i>Optional.</i> When using the <i>markerObjectClass</i> keyword to specify the scope of the uniqueness check instead of a DN, it is also possible to specify to perform the check only if the updated entry contains the objectclass specified in the <i>requiredObjectClass</i> keyword. Do not include a space before or after the equals sign.

Table 18.1. Attribute Uniqueness Plug-in Variables

3. Creating an Instance of the Attribute Uniqueness Plug-in

To ensure that a particular attribute in the directory always has unique values, create an instance of the Attribute Uniqueness Plug-in for the attribute to check. For example, to ensure that every entry in the directory that includes a *mail* attribute has a unique value for that attribute, create a mail uniqueness plug-in.

To create an instance of the Attribute Uniqueness Plug-in, modify the Directory Server configuration to add an entry for the new plug-in under the *cn=plugins,cn=config* entry. The format of the new entry must conform to the syntax described in [Section 2, “Attribute Uniqueness Plug-in Syntax”](#).

**NOTE**

Red Hat strongly encourages you to copy and paste an existing Attribute Uniqueness Plug-in entry and only modify the attributes listed below.

For example, to create an instance the Attribute Uniqueness Plug-in for the mail attribute, do the following:

1. In the `dse.ldif` file, locate the entry for the Attribute Uniqueness Plug-in, `cn=attribute uniqueness,cn=plugins,cn=config`.
2. Copy the entire entry. The entry ends in an empty line; copy the empty line, too.
3. Paste the copied Attribute Uniqueness Plug-in entry at the end of the file.
4. Modify the Attribute Uniqueness Plug-in entry attributes for the new attribute information:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=example,dc=com
...
```

5. Restart the Directory Server.

```
service dirsrv restart instance_name
```

In this example, the uniqueness check will be performed on every entry in the `dc=example,dc=com` entry that includes the `mail` attribute.

4. Configuring Attribute Uniqueness Plug-ins

This section explains how to use Directory Server Console to view the plug-ins configured for the directory and how to modify the configuration of the Attribute Uniqueness Plug-ins.

4.1. Viewing Plug-in Configuration Information

Display the configuration entry for Attribute Uniqueness Plug-ins as follows:

1. In the Directory Server Console, click the **Directory** tab.
2. In the left navigation tree, expand the **config** folder, then the **Plug-ins** folder.

The list of plug-ins is displayed in the right navigation window, which shows the UID Attribute Uniqueness Plug-in and any other Attribute Uniqueness Plug-ins that have been created. (See [Section 3, “Creating an Instance of the Attribute Uniqueness Plug-in”](#).)

3. In the right navigation window, double-click the plug-in entry to view.

The **Property Editor** opens. It contains a list of all the attributes and values for the plug-in.

4.2. Configuring Attribute Uniqueness Plug-ins from the Directory Server Console

The plug-in configuration can be updated from the Directory Server Console in several ways:

- From the **Property Editor**.
 1. In the Directory Server Console, click the **Directory** tab.
 2. In the left navigation tree, expand the **config** folder, then the **Plug-ins** folder.
 3. Select the plug-in instance.
 4. Edit the attribute value fields.
- From the **Configuration** tab.
 1. In the Directory Server Console, select the **Configuration** tab; then, in the navigation tree, expand the **Plug-ins** folder, and select the Attribute Uniqueness Plug-in to modify.

The configuration parameters for the plug-in are displayed in the right pane.

2. To turn the plug-in on or off, check or clear the **Enable Plugin** checkbox.
3. To add a suffix or subtree, click **Add**, and type a DN in the blank text field.

To avoid using a DN, enter the `markerObjectClass` keyword. With this syntax, it is possible to click **Add** again to specify a `requiredObjectClass`, as described in [Section 2, “Attribute Uniqueness Plug-in Syntax”](#).



NOTE

Do *not* add an attribute name to the list. To check the uniqueness of other attributes, create a new instance of the Attribute Uniqueness Plug-in for the attribute to check. For information, see [Section 3, “Creating an Instance of the Attribute Uniqueness Plug-in”](#).

4. To delete an item from the list, place the cursor in the text field to delete, and click **Delete**.

5. Click **Save**.

4.3. Configuring Attribute Uniqueness Plug-ins from the Command-Line

This section provides information about configuring the plug-in from the command line.

- [Section 4.3.1, “Turning the Plug-in On or Off”](#)
- [Section 4.3.2, “Specifying a Suffix or Subtree”](#)
- [Section 4.3.3, “Using the `markerObjectClass` and `requiredObjectClass` Keywords”](#)

4.3.1. Turning the Plug-in On or Off

1. To turn the plug-in on from the command line, run `ldapmodify` using an LDIF update statement to change the `nsslapd-pluginenabled` attribute. For example:

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

dn: cn=descriptive_plugin_name,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

For detailed information on the `ldapmodify` command, see the *Directory Server Configuration, Command, and File Reference*.

To disable the plug-in, change the LDIF update statements to replace the `on` with `off`.

2. Whenever a plug-in is enabled or disabled, the server must be restarted.

```
service dirsrv restart instance_name
```

For information on restarting the server, see [Section 3, “Starting and Stopping Servers”](#).

4.3.2. Specifying a Suffix or Subtree

The suffix or subtrees which the plug-in checks to ensure attribute uniqueness are defined using the `nsslapd-pluginarg` attribute in the entry defining the plug-in.

To specify the subtree or subtrees, use `ldapmodify` to send LDIF update statements, similar to this example:

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

```
dn: cn=mail uniqueness,cn=plugins,cn=config
changetype: modify
add: nsslapd-pluginarg2 nsslapd-pluginarg3
nsslapd-pluginarg2: ou=Engineering,dc=example,dc=com
nsslapd-pluginarg3: ou=Sales,dc=example,dc=com
```

This example LDIF statement modified the Attribute Uniqueness Plug-in to check the uniqueness of the *mail* attribute under the subtrees *dc=example,dc=com*, *ou=Engineering,dc=example,dc=com*, and *ou=Sales,dc=example,dc=com*.

Use the `ldapmodify` command to import the LDIF file into the directory. For detailed information on the `ldapmodify` command, see the *Directory Server Configuration, Command, and File Reference*.

Whenever this type of configuration change is made, restart the server.

```
service dirsrv restart instance_name
```

For information on restarting the server, see [Section 3, “Starting and Stopping Servers”](#).

4.3.3. Using the `markerObjectClass` and `requiredObjectClass` Keywords

Instead of specifying a suffix or subtree in the configuration of an Attribute Uniqueness Plug-in, perform the check under the entry belonging to the DN of the updated entry that has the object class given in the `markerObjectClass` keyword.

To specify to perform the uniqueness check under the entry in the DN of the updated entry that contains the organizational unit (*ou*) object class, copy and paste an existing Attribute Uniqueness Plug-in entry, and change the following attributes:

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

dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=ou
...
```

If the server should not check every entry in the organization unit, limit the scope by setting the check to be performed only if the updated entry contains a specified object class.

For example, if the uniqueness of the *mail* attribute is checked, it is probably only necessary to perform the check when adding or modifying entries with the *person* or *inetorgperson* object class.

Restrict the scope of the check by using the `requiredObjectClass` keyword, as shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=ou
nsslapd-pluginarg2: requiredObjectClass=person
...
```

The `markerObjectClass` or `requiredObjectClass` keywords *cannot* be repeated by incrementing the counter in the `nsslapd-pluginarg` attribute suffix. These keywords can only be used once per Attribute Uniqueness Plug-in instance.



NOTE

The `nsslapd-pluginarg0` attribute always contains the name of the attribute for which to ensure uniqueness.

5. Attribute Uniqueness Plug-in Syntax Examples

This section contains examples of Attribute Uniqueness Plug-in syntax in the `dse.ldif` file.

- [Section 5.1, “Specifying One Attribute and One Subtree”](#)
- [Section 5.2, “Specifying One Attribute and Multiple Subtrees”](#)

5.1. Specifying One Attribute and One Subtree

This example configures the plug-in to ensure the uniqueness of the `mail` attribute under the `dc=example,dc=com` subtree.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=example,dc=com
...
```

5.2. Specifying One Attribute and Multiple Subtrees

It is possible use a single plug-in instance to check for the uniqueness of an attribute within multiple subtrees, which means that the attribute value must be unique *within* each subtree but

not unique across all subtrees. This example configures the Attribute Uniqueness Plug-in to ensure the uniqueness of the *mail* attribute for separate subtrees, *l=Chicago,dc=example,dc=com* and *l=Boston,dc=example,dc=com*.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: l=Chicago,dc=example,dc=com
nsslapd-pluginarg2: l=Boston,dc=example,dc=com
...
```



NOTE

The *nsslapd-pluginarg0* attribute always contains the name of the attribute for which to ensure uniqueness. All other occurrences of the *nsslapd-pluginarg*, such as *nsslapd-pluginarg1*, contain DNs.

With this configuration, the plug-in allows an instance of a value for the *mail* attribute to exist once under the *l=Chicago,dc=example,dc=com* subtree and once under the *l=Boston,dc=example,dc=com* subtree. For example, the following two attribute-value settings are allowed:

```
mail=bjensen,l=Chicago,dc=example,dc=com
mail=bjensen,l=Boston,dc=example,dc=com
```

To ensure that only one instance of a value exists under both subtrees, configure the plug-in to ensure uniqueness for the entire *dc=example,dc=com* subtree.

6. Replication and the Attribute Uniqueness Plug-in

When using the Attribute Uniqueness Plug-ins on Directory Servers involved in a replication agreement, carefully consider how to configure the plug-in on each server.

Consider the following cases:

- Simple replication with one supplier and one or several consumers.
- Complex replication with multiple masters.

Attribute Uniqueness Plug-ins do not perform any checking on attribute values when an update is performed as part of a replication operation.

6.1. Simple Replication Scenario

Because all modifications by client applications are performed on the supplier server, the Attribute Uniqueness Plug-in should be enabled on the supplier. It is unnecessary to enable it on the consumer server.

Enabling the Attribute Uniqueness Plug-in on the consumer does not prevent Directory Server from operating correctly but is likely to cause a performance degradation.

6.2. Multi-Master Replication Scenario

In a multi-master replication scenario, the masters act both as suppliers and consumers of the same replica. Because multi-master replication uses a loosely consistent replication model, enabling an Attribute Uniqueness Plug-in on one of the servers is not sufficient to ensure that attribute values will be unique across both supplier servers at any given time. Therefore, enabling an Attribute Uniqueness Plug-in on one server can cause inconsistencies in the data held on each replica.

However, it is possible to use an Attribute Uniqueness Plug-in, providing both of the following conditions are met:

- The attribute on which the uniqueness check is performed is a naming attribute.
- The Attribute Uniqueness Plug-in is enabled on both supplier servers.

When these conditions are met, attribute uniqueness conflicts are reported as naming conflicts at replication time. Naming conflicts require manual resolution. For information on how to resolve replication conflicts, see [Section 18, “Solving Common Replication Conflicts”](#).

Synchronizing Red Hat Directory Server with Microsoft Active Directory

The Windows Sync feature allows synchronization of adds, deletes, and changes in groups, users, and passwords between Red Hat Directory Server and Microsoft Active Directory. It provides an efficient and effective way to maintain consistent information across directories.

1. About Windows Sync

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 Sync has two parts, the sync service for directory entries and the sync service for passwords:

- *Directory Server Windows Sync.* The Directory Server leverages the Multi-Master Replication Plug-in to synchronize user and group entries. The same changelog that is used for multi-master replication is also used to send updates from the Directory Server to Active Directory as an LDAP operation. The server also performs LDAP search operations against its Windows server to synchronize changes made to Windows entries to the corresponding Directory Server entry. This is illustrated in [Figure 19.1, “Active Directory - Directory Server Synchronization Process”](#).



Figure 19.1. Active Directory - Directory Server Synchronization Process

- *Password Sync Service*. This application captures password changes for Windows users and relays those changes back to the Directory Server over LDAPS. It must be installed on the Active Directory machine. This is done separately from the Windows Sync service to accommodate password encryption.

Synchronization is configured and controlled by one or more *synchronization agreements*, which establishes synchronization between *sync peers*, the directory servers being synced. These are similar in purpose to replication agreements and contain a similar set of information, including the hostname and port number for Active Directory. The Directory Server connects to its peer Windows server via LDAP/LDAPS to both send and receive updates.

A single Active Directory subtree is synchronized with a single Directory Server subtree, and vice versa. Unlike replication, which connects *databases*, synchronization is between *suffixes*, parts of the directory tree structure. The synced 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 Sync 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 Sync 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. Because there is no changelog on the Active Directory side, the **Dirsync** search is issued periodically, every five minutes. 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 3.5, “Manually Updating and Resynchronizing Entries”](#) for more information.

Windows Sync 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 (`nsDS7NewWinUserSync` and `nsDS7NewWinGroupSync`) 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.

See [Section 3, “Using Windows Sync”](#) for more information on creating user and group entries.

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 master in a multi-master replication environment, and use that master to replicate the changes across the Directory Server deployment, as shown in [Figure 19.2, “Multi-Master Directory Server - Windows Domain Synchronization”](#).



CAUTION

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.

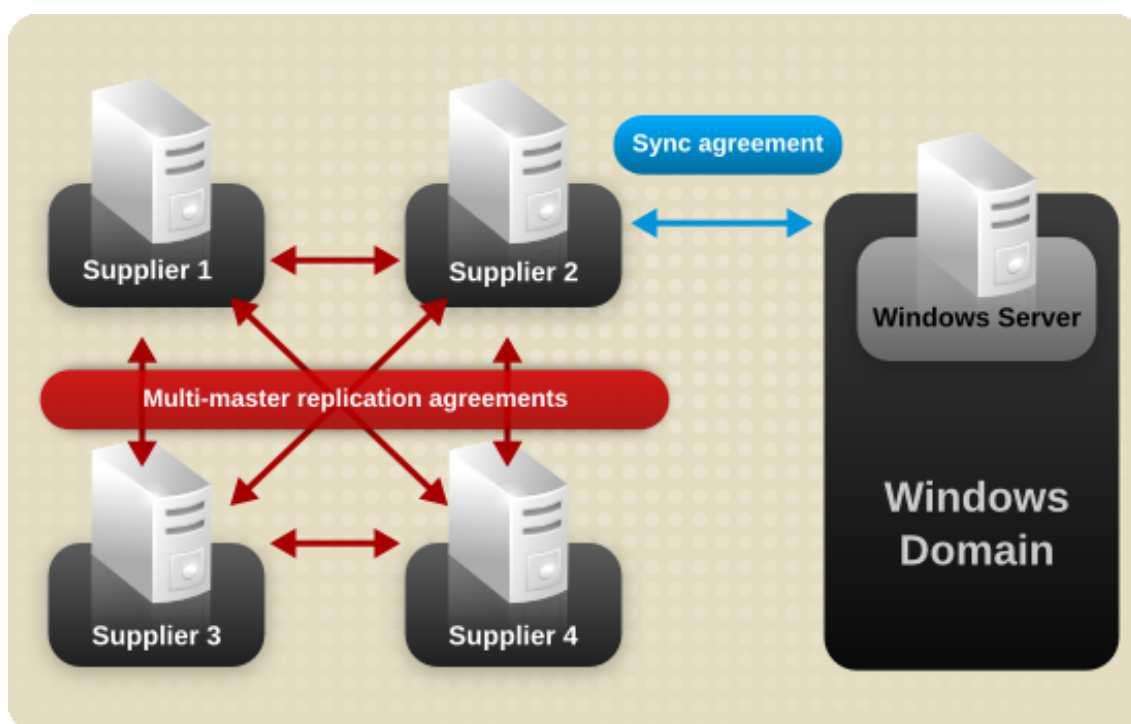


Figure 19.2. Multi-Master Directory Server - Windows Domain Synchronization

Directory Server passwords are synchronized along with other entry attributes because plain-text passwords are retained in the Directory Server changelog. The **Password Sync** Service is needed to catch password changes made on Active Directory. Without the **Password Sync** 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.

2. Configuring Windows Sync

2.1. Step 1: Configure SSL on Directory Server

To configure the Directory Server to run in SSL, see [Chapter 11, Managing SSL](#). To configure SSL on Active Directory, see the appropriate user documentation.

Use the `certutil` utility to create self-signed certificates or obtain and install certificates to enable SSL; for more information, see [Section 3, "Using certutil"](#).

The following certificates must be issued and installed on both the Directory Server and the Active Directory sync peer:

- CA certificate, shared between the Directory Server and Active Directory

- Directory Server certificate, accessible by the sync services

2.2. Step 2: Configure the Active Directory Domain

The Active Directory domain has to be properly configured for synchronization to work.

1. Set up the Windows domain. On Windows 2000, use the `dcpromo` tool. On Windows 2003, install the domain controller for Active Directory by clicking **Add or Remove Programs** and then **Add/Remove Windows Components**.



NOTE

For more detailed information, see the appropriate Windows documentation.

2. Make sure that the Active Directory password complexity policies are enabled so that the **Password Sync** service will run.

Run `secpol.msc`, and select **Security Settings**, then **Account Policies**, and **Password Policy**. Make sure that `Password must meet complexity requirements` is selected.

3. Set up SSL on the Active Directory server.
 - a. Install a certificate authority in the **Windows Components** section in **Add/Remove Programs**.
 - b. Select the **Enterprise Root CA** option.
 - c. Reboot the Active Directory server. If IIS web services are running, the CA certificate can be accessed by opening `http://servername/certsrv`.
 - d. Set up the Active Directory server to use the SSL server cert.
 - i. Create a certificate request `.inf`, using the fully-qualified domain name of the Active Directory as the certificate subject.
 - ii. Request the certificate by running the following command on the Active Directory machine:

```
certreq -new request.inf request.req
```

- iii. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```



NOTE

If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is `http://servername/certsrv`.

iv. Accept the certificate request. For example:

```
certreq -accept cernew.cer
```

v. Make sure that the server certificate is present on the Active Directory server. In the **File** menu, click **Add/Remove**, then click **Certificates** and **Personal>Certificates**.

vi. Import the CA certificate from Directory Server into Active Directory. Click **Trusted Root CA**, then **Import**, and browse for the Directory Server CA certificate.

For more information, see <http://support.microsoft.com/default.aspx?scid=kb;en-us;321051>.

2.3. Step 3: Select or Create the Sync Identity

There are two users used to configure Windows Sync: an Active Directory user, specified in the sync agreement, and a Directory Server user, specified in the **Password Sync** service.

The user specified in the sync agreement is the entity as whom the Directory Server binds to Active Directory to send and receive updates. The Active Directory user should be a member of the Domain Admins group, or have equivalent rights, and must have rights to replicate directory changes. This limits the extent of the Windows directory that can be affected by the sync ID to only the synchronized subtree. For information on adding users and setting privileges in Active Directory, see the Microsoft documentation.

The user references in the **Password Sync** service must have read and write permissions to every entry within the synchronized subtree and absolutely must have write access to password attributes in Directory Server so that **Password Sync** can update password changes.

For security reasons, the **Password Sync** user should not be Directory Manager and should not be part of the synchronized subtree. For information on adding users, see [Chapter 2, Creating Directory Entries](#); for information on setting permissions, see [Chapter 6, Managing Access Control](#). For information on creating a special sync ID, see [Section 3, “Creating the Supplier Bind DN Entry”](#)

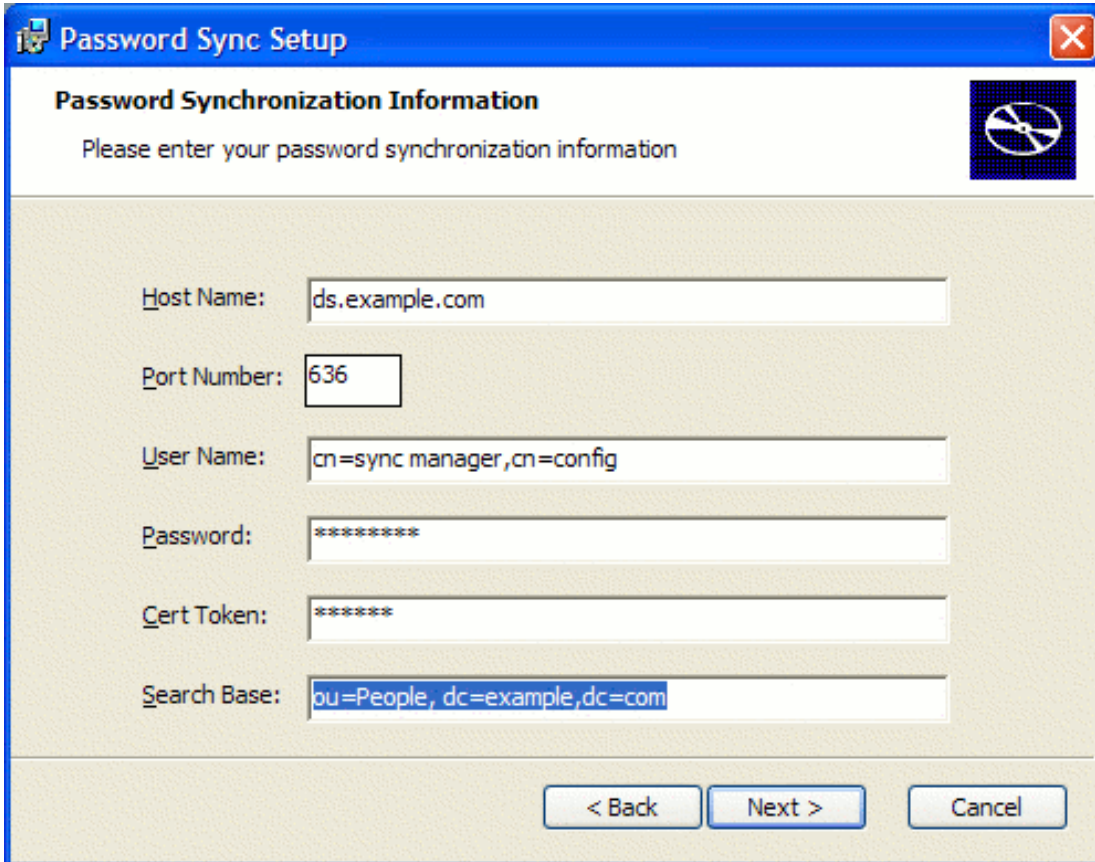
**NOTE**

The user cited in the sync agreement (the supplier DN) exists on the Active Directory server. The user cited in the **Password Sync** configuration exists on Directory Server.

2.4. Step 4: Install and Configure the Password Sync Service

Password Sync can be installed on any Windows machine to synchronize Windows passwords. Passwords can only be synchronized if both the Directory Server and Windows server are running in SSL, the sync agreement is configured over an SSL connection, and certificate databases are configured for **Password Sync** to access.

1. Copy the `PassSync.msi` file that contains the **Password Sync** utility to the Active Directory machine.
2. Double-click on the `PassSync.msi` file to install it.
3. The **Password Sync** Setup window will appear. Hit **Next** to begin installing.
4. Fill in the Directory Server hostname, secure port number, user name (such as `cn=syncmanager,cn=config`), the certificate token (password), and the search base (e.g., `ou=People,dc=example,dc=com`).



The image shows a Windows-style dialog box titled "Password Sync Setup". It has a blue title bar with a close button (X) in the top right corner. Below the title bar, the text "Password Synchronization Information" is displayed in bold, followed by the instruction "Please enter your password synchronization information". To the right of this text is a small icon of a hard drive with a diagonal line through it. The main area of the dialog contains several input fields: "Host Name:" with the value "ds.example.com", "Port Number:" with the value "636", "User Name:" with the value "cn=sync manager,cn=config", "Password:" with masked characters "*****", "Cert Token:" with masked characters "*****", and "Search Base:" with the value "ou=People, dc=example, dc=com". At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Figure 19.3. Setting up Password Sync Information

Hit **Next**, then **Finish** to install **Password Sync**.

5. Reboot the Windows machine to start **Password Sync**.



NOTE

The Windows machine must be rebooted. Without the rebooting, `PasswordHook.dll` will not be enabled, and password synchronization will not function.

Password Sync is installed in `C:\Program Files\Red Hat Directory Password Synchronization`.

The following DLLs are installed in `C:\winnt\system32` and utilized by **Password Sync**:

`passhook.dll`
`nsldap32v50.dll`

nsldapssl32v50.dll
libplc4.dll
nsldappr32v50.dll
nss3.dll
libnspr4.dll
ssl3.dll
libplds4.dll
softokn3.dll

Next, set up certificates that **Password Sync** will use to access the Directory Server over SSL:



NOTE

SSL is required for **Password Sync** to send password to Directory Server. The service will not send the passwords except over SSL to protect the clear text password sent from the Active Directory machine to the Directory Server machine.

1. Download `certutil.exe` if it is not already installed on the machine. It is available from <ftp://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/>. See *Chapter 11, Managing SSL* for more information on SSL.
2. Create a new `cert8.db` and `key.db` using `certutil.exe` on the **Password Sync** machine.

```
certutil.exe -d . -N  
ln -s slapd-serverID-cert8.db cert8.db  
ln -s slapd-serverID-key3.db key3.db
```

3. On the Directory Server, export the server certificate using `pk12util`.

```
pk12util -d . -o  
servercert.pfx -n Server-Cert
```

4. Copy the exported certificate from the Directory Server to the Windows machine.
5. Import the server certificate from the Directory Server into the new certificate databases using `pk12util.exe`.

```
pk12util.exe -d "C:\Program Files\Red Hat Directory Password  
Synchronization" -i servercert.pfx
```

6. Give trusted peer status to the server.

```
certutil.exe -d "C:\Program Files\Red Hat Directory Password  
Synchronization" -M  
-n Server-Cert -t "P,P,P"
```



NOTE

If any Active Directory user accounts exist when **Password Sync** is first installed, then the passwords for those user accounts cannot be synchronized until they are changed because **Password Sync** cannot decrypt a password once it has been hashed in Active Directory.

2.5. Step 5: Configure the Directory Server Database for Synchronization

Just as with replication, there must be a changelog available to track and send directory changes and the Directory Server database being synchronized must be configured as a replica.



NOTE

If the Directory Server database is already in a replicated environment, this step is not necessary.

First, enable the changelog:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click the **Replication** folder.
3. In the main window, click the **Supplier Settings** tab.
4. Check the **Enable Changelog** database.
5. Set the changelog database directory. Click the **Use default** button to use the default or **Browse...** to select a custom directory.
6. Save the changelog settings.

After setting up the changelog, then configure the database that will be synchronized as a

replica. The replica role should be either a single-master or multi-master.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click the **Replication** folder, then click the name of the database to synchronize.

By default, there are two databases, `NetscapeRoot` for directory configuration and `userRoot` for directory entries. Other databases may be listed if they have been added to Directory Server.

3. Check the **Enable Replica** checkbox, and select the radio button by the type of replica which the database will be.
4. In the **Update Settings** section, either select or add a supplier DN. This is the user account as which synchronization process will be run. As mentioned in [Section 2.3, “Step 3: Select or Create the Sync Identity”](#), this user must be on the Active Directory server.
5. Save the replication settings for the database.



NOTE

For more information on replication settings, see [Chapter 8, Managing Replication](#).

2.6. Step 6: Create the Synchronization Agreement

Create the synchronization agreement:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click **Replication**, then right-click on the database to sync. The default user database is `userRoot`, but additional databases are added as new suffuxes are added to the Directory Server.

Alternatively, highlight the database, and in the top tool bar, click **Object**.

3. Select **New Windows Sync Agreement** from the menu.


This opens the **Synchronization Agreement Wizard**.

4. In the two fields, supply a name and description of the synchronization agreement. Hit **Next**.
5. The second screen reads **Windows Sync Server Info**. By default, the Directory Server hostname and port are visible at the top, under Supplier. At the very bottom of the screen, the

name of the synced suffix, such as `dc=example,dc=com`, is displayed.

Windows Sync Server Info

Provide server and content information:

Supplier:  example.com: 636

Windows Domain Information:

Windows Domain Name:

Sync New Windows Users: ☒ Sync New Windows Groups: ☒

Windows Subtree:

DS Subtree:

Domain Controller Host:

Port Num:

Connection:

☒ Using encrypted SSL connection

Bind as:

Password:

Subtree:

Back Next Cancel Help

Figure 19.4. Setting up the Sync Agreement

6. In the middle of the screen are fields for the Windows domain information. Fill in the domain name and the domain controller.
7. Select the checkboxes for the Windows entries which are going to be synchronized.

- *Sync New Windows Users*. When enabled, all user entries found in Windows that are subject to the agreement will automatically be created in the Directory Server.
 - *Sync New Windows Groups*. When enabled, all group entries found in Windows that are subject to the agreement will automatically be created in the Directory Server.
8. The Windows and Directory Server subtree information is automatically filled in; use the defaults to sync only users or change these as appropriate to sync groups or groups and users.
 9. Check the **Using encrypted SSL connection** checkbox. The use of SSL is recommended for security reasons, and SSL is required for synchronizing passwords because Active Directory will refuse to modify passwords unless the connection is SSL-protected.
 10. Fill in the authentication information in the **Bind as...** and **Password** fields with the sync ID information. This user must be on both the Active Directory server and will be one of the supplier DN's available in the database replication setup, as described in [Section 2.5, "Step 5: Configure the Directory Server Database for Synchronization"](#).
 11. The last screen is a summary of the synchronization agreement. It is possible to modify all of the configuration at this using the back buttons to get to the appropriate screen. If the agreement is correct, click **Done**.

When the agreement is complete, an icon representing the synchronization agreement is displayed under the suffix. This icon indicates that the synchronization agreement is set up.

2.7. Step 7: Begin Synchronization

After the sync agreement is created, begin the synchronization process. Select the sync agreement, right-click or open the **Object** menu, and select **Begin resynchronization**. This will begin the synchronization process.

If synchronization stops for any reason, begin another total update (resynchronization) by selecting this from the sync agreement menu. Beginning a total update (resynchronization) will not delete or overwrite the databases.

3. Using Windows Sync

After the sync agreement is setup, synchronize the user and group entries on the Directory Server and Active Directory server.

- [Section 3.1, "Synchronizing Users"](#)
- [Section 3.2, "Synchronizing Groups"](#)
- [Section 3.3, "Deleting Entries"](#)

- [Section 3.5, “Manually Updating and Resynchronizing Entries”](#)
- [Section 3.6, “Checking Synchronization Status”](#)
- [Section 3.7, “Modifying the Sync Agreement”](#)

3.1. Synchronizing Users

If Windows users are synchronized when the sync agreement was created, all the existing Windows users are synchronized to the Directory Server after the first total update (when synchronization begins). When a new Windows user account is created, a corresponding entry will automatically be created on the peer Directory Server. If an existing sync agreement is modified to begin synchronizing users, the Windows users will be added to the Directory Server after the next total update.

A new Directory Server user account is synchronized to a Windows server if the new Directory Server entry uses the `ntUser` object class and the `ntUserCreateNewAccount` attribute. New users that are created on the Directory Server with the `ntUser` object class are synced to the Windows machine at the next regular update; existing users that have the `ntUser` object class added are synchronized at the next total update.

Special schema are applied to synchronized user entries in the Directory Server. This schema are similar, but not identical, to that used by Netscape Directory Server 4.x NT Synchronization.

All synchronized entries in the Directory Server, whether they originated in the Directory Server or in Active Directory, have special synchronization attributes.

- *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.
- *ntDomainUser*. This corresponds to the `samAccountName` attribute for Active Directory entries.
- *ntUserDeleteAccount*. This attribute is set automatically when a Windows entry is synced 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.

Setting `ntUserCreateNewAccount` and `ntUserDeleteNewAccount` on Directory Server entries allows the Directory Manager fine-grained control over which users within the synchronized subtree will be synced on Active Directory, similar to selecting in the sync agreement whether to synchronize new Windows users.

When creating a Directory Server user in the Console (see [Section 1.2, “Creating Directory Entries”](#)), there is an **NT User** tab in the **New User** dialog. Fill in this information to supply Windows attributes automatically.

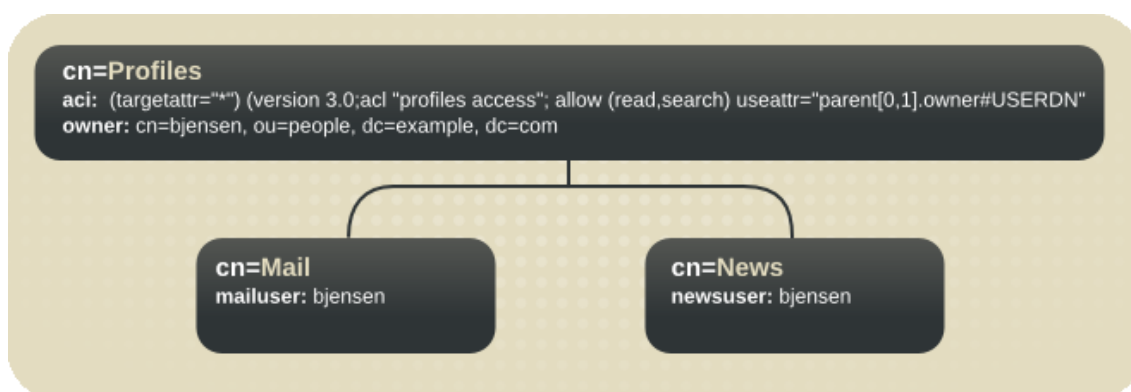


Figure 19.5. Setting User Attributes

Additional `ntUser` attributes can be created either by using the **Advanced** button in the Console or by using `ldapmodify`; see [Section 2.4.2, “Modifying Entries Using Ldapmodify”](#).

[Table 19.1, “User Schema Mapped between Directory Server and Active Directory”](#) shows the attributes that are mapped between the Directory Server and Windows servers, and [Table 19.2, “User Schema That Are the Same in Directory Server and Windows Servers”](#) shows the attributes that are the same between the Directory Server and Windows servers. For more information on the interaction between Directory Server and Windows schema, see [Section 4, “Schema Differences”](#).

Directory Server	Active Directory
cn	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

Table 19.1. User Schema Mapped between Directory Server and Active Directory

cn	physicalDeliveryOfficeName
description	postOfficeBox
destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenName	registeredAddress
homePhone	sn
homePostalAddress	st
initials	street
l	telephoneNumber
mail	teletexTerminalIdentifier
manager	telexNumber
mobile	title
o	userCertificate
ou	x121Address
pager	

Table 19.2. User Schema That Are the Same in Directory Server and Windows Servers

3.2. Synchronizing Groups

All existing Windows groups are synchronized to the Directory Server during the first total update (when synchronization first begins). When a new Windows group is created, a corresponding entry is automatically created on the peer Directory Server if that option is selected in the sync agreement.

Similar to user entries, Directory Server group entries are synchronized if they have the `ntGroup` object class.

Like with Directory Server entries, there are two attributes that control creation and deletion of entries in Active Directory, `ntGroupCreateNewAccount` and `ntGroupDeleteAccount`.

Additionally, groups have the following two attributes:

- *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.
- *ntGroupType*. This is set automatically for Windows groups that are synchronized over, but this attribute must be set manually on Directory Server entries before they will be synched.

The membership of groups is synchronized with the constraint that only those members that are also within the scope of the agreement are propagated. Group members that are not within the scope of the agreement are left unchanged on both sides.

Table 19.3, “Group Entry Attribute Mapping between Directory Server and Active Directory” shows the attributes that are mapped between the Directory Server and Windows servers, and *Table 19.4, “Group Entry Attributes That Are the Same between Directory Server and Active Directory”* shows the attributes that are the same between the Directory Server and Windows servers.

Directory Server	Active Directory
cn	name
ntGroupAttributes	groupAttributes
ntGroupId	cn name samAccountName
ntGroupType	groupType

Table 19.3. Group Entry Attribute Mapping between Directory Server and Active Directory

cn	member
description	ou
l	seeAlso

Table 19.4. Group Entry Attributes That Are the Same between Directory Server and Active Directory

3.3. Deleting Entries

An Active Directory group or user account is automatically deleted from the Directory Server sync peer server when entry is deleted. The same is true when a Directory Server account is

deleted if the deleted entry has the `ntUserDeleteAccount` or `ntGroupDeleteAccount` 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 five minutes so that the `ntUniqueId` attribute is synchronized, and then delete the entry.

3.4. 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 synched between Directory Server and Active Directory is re-added to Directory Server, the resurrected Directory Server 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. The way that tombstone entries are handled is different between Windows Server 2000 and Windows Server 2003:

- On Windows 2000, Active Directory creates a new entry with a new unique ID; this new ID is synched back to the Directory Server entry.
- On Windows 2003, Active Directory resurrects the old entry and preserves the original unique ID for the entry.

For Active Directory entries on both on Windows 2000 and 2003, 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.

3.5. Manually Updating and Resynchronizing Entries

Synchronization occurs every five minutes. However, an incremental update can be done manually if there are changes that need synchronized immediately.

To perform an incremental update manually:

1. Go to the **Configuration** tab in the Console.
2. Right-click on the synchronization agreement icon, and select **Send and Receive Updates** from the drop down menu.

During normal operations, all the updates made to entries in the Directory Server that need to be sent to Active Directory are collected the changelog and then replayed during an incremental update.

However, when the synchronization is initially configured, 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.

To send a total update:

1. Go to the **Configuration** tab in the Console.
2. Right-click on the synchronization agreement icon, and select **Initialize Re-synchronization** from the drop down menu.

This will not delete data on the sync peer; it will send and receive all updates and add any new or modified Directory Server entries; for example, it will add a pre-existing Directory Server user that had the `ntUser` object class added.

3.6. Checking Synchronization Status

Check synchronization status in the **Replication** tab in the **Status** of the Console. Highlight the synchronization agreement to monitor, and the relevant information should appear in the right-hand pane. The **Status** area shows whether the last incremental and total updates were successful and when they occurred.

3.7. Modifying the Sync Agreement

It is possible to modify parts of the synchronization agreement after it has been created.

In the **Configuration>Replication** tab of the Directory Server Console, select the sync agreement icon from beneath the database. There are two tabs, **Summary** and **Connection**.

- The **Summary** tab allows the description of the agreement to be changed. This tab also shows the sync peer host and port information and synchronized subtrees.

- The **Connection** tab allows the bind DN and bind credentials for the sync ID to be changed and shows whether Windows users and groups are synchronized. It also shows whether synchronization occurs over an SSL connection.

4. Schema Differences

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.

4.1. Password Policies

Both Active Directory and Directory Server can enforce password policies such as password minimum length or maximum age. Windows Sync 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 synched to the other system. The default password syntax setting on Directory Server mimics the default password complexity rules that Active Directory enforces.

4.2. Groups

Nested groups (where a group contains another group as a member) are supported and for WinSync will be synchronized. However, Active Directory imposes certain constraints as to the composition of nested groups. For example, a global group 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.

4.3. 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 Sync 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 synced 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.

4.4. 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.

5. Password Sync Service

The **Password Sync** service must be installed on the Active Directory server. It synchronizes password changes made on Active Directory with the corresponding entries' passwords on the Directory Server. Like any Windows service, it can be modified, started and stopped, and uninstalled, depending on how synchronization between Directory Server and Active Directory changes.

5.1. Modifying Password Sync

To reconfigure **Password Sync**, open the Windows Services panel, highlight **Password Sync**, and select **Modify**. This goes back through the configuration screens.

5.2. Starting and Stopping the Password Sync Service

The **Password Sync** service is configured to start whenever the Active Directory host is started. To reconfigure the service so that it does not start when Windows reboots:

1. Go to the **Control Panel**, and select **Services**.
2. Scroll through the list of services for the **Password Sync** service. The **Startup** field is set to `Automatic`.
3. Double-click on **Password Sync**.
4. Select the **Manual** radio button, and then click **OK**.

To start and stop **Password Sync**, do the following:

1. Go to the **Control Panel**, and select **Services**.

2. Scroll through the list of services for **Password Sync**, and right-click on it.
3. Select **Stop** or **Start**, and hit okay.

Changed passwords are captured even if **Password Sync** is not running. If **Password Sync** is restarted, the password changes are sent to Directory Server at the next synchronization.

5.3. Uninstalling Password Sync Service

To uninstall the **Password Sync** service, do the following:

1. Open the **Add/Remove Programs** utility.
2. Select click remove to uninstall the **Password Sync** service.
3. If SSL was configured for the **Password Sync**, then the `cert8.db` and `key3.db` databases that were created were not removed when **Password Sync** was uninstalled. Delete these files by hand.

6. Troubleshooting

If synchronization does not seem to be functioning properly, see the Windows event log and/or Directory Server error log for information on any potential problems.

Enable replication logging for more detailed information on synchronization to be recorded in the error logs. Replication log levels will produce more verbose logs from the sync code that can help in diagnosing problems.

1. In the Console, click the **Configuration** tab, select **Logs** from the navigation menu on the right, and open the error log.
2. Scroll down to error log level, and select **Replication** from the menu. Hit save.

For complete information on error log levels, refer to *Red Hat Directory Server Configuration, Command, and File Reference*.

Error #1: The message box when creating the sync agreement indicates that the it cannot connect to Active Directory.

Make sure that the directory suffixes, Windows domain and domain host, and the administrator DN and password are correct. Also verify that the port numbers used for LDAPS is correct. If all of this is correct, make sure that Active Directory or the Windows machine are running.

Error #2: After synchronization, the status returns error 81.

One of the sync peer servers has not been properly configured for SSL communication. Examine the Directory Server access log file to see if the connection attempt was received by

the Directory Server. 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 (port number, hostname, search base, and so forth) to see if any of these are the problem. If all else fails, reconfigure the Directory Server with a new certificate.

If the LDAPS connection is successful, it is likely that the misconfiguration is on 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.

Appendix A. LDAP Data Interchange Format

Red Hat Directory Server (Directory Server) uses the LDAP Data Interchange Format (LDIF) to describe a directory and directory entries in text format. LDIF is commonly used to build the initial directory database or to add large numbers of entries to the directory all at once. In addition, LDIF is also used to describe changes to directory entries. For this reason, most of Directory Server's command-line utilities rely on LDIF for either input or output.

Because LDIF is a text file format, 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 2, Creating Directory Entries](#).

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 A.1, “LDIF Fields”](#) describes the LDIF fields shown in the previous definition.

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.
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 Chapter 9, Extending the Directory Schema 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 Chapter 9, Extending the Directory Schema 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 information on attribute subtypes, see Section 1.3.8, “Adding an Attribute Subtype” . For a complete list of the supported subtypes tags, see Table D.2, “Supported Language Subtypes” .
<i>attribute_value</i>	Specifies the attribute value to be used with the attribute type.

Table A.1. LDIF Fields



NOTE

The LDIF syntax for representing a change to an entry in the directory is different from the syntax described in [Table A.1, “LDIF Fields”](#). For information on using LDIF to modify directory entries, see [Chapter 2, Creating Directory Entries](#).

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=Jake Lupinski,dc=example,dc=com

dn: cn=Jake Lup
   inski, dc=exa
   mple,dc=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.

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.

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 -D userDN -w user_password

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

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 SSL 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.

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.

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 A.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 <code>top</code> object class.
objectClass: domain	Specifies the <code>domain</code> object class. This line defines the entry as a domain or domain component.
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 <code>dc=hostname,dc=domain,dc=toplevel</code> . For example, <code>dc=ldap,dc=example,dc=com</code> . The domain entry should use the leftmost <code>dc</code> value, such as <code>dc: ldap</code> . If the suffix were <code>dc=example,dc=com</code> , the <code>dc</code> value is <code>dc: example</code> . Do not create the entry for <code>dn: dc=com</code> 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.

Table A.2. LDIF Elements in Domain Entries

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 A.3, “LDIF Elements in Organizational Unit Entries” defines each element of the LDIF-formatted organizational unit entry.

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 <code>dn: ou=people,dc=example,dc=com</code> .
objectClass: top	<i>Required.</i> Specifies the <code>top</code> object class.
objectClass: organizationalUnit	Specifies the <code>organizationalUnit</code> object class. This line defines the entry as an organizational unit.
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.

Table A.3. LDIF Elements in Organizational Unit Entries

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 A.4, “LDIF Elements in Person Entries” defines each aspect of the LDIF person entry.

LDIF Element	Description
dn: <i>distinguished_name</i>	<i>Required.</i> Specifies the distinguished name for the entry. For example, dn: uid=bjensen,ou=people,dc=example,dc=com. If there is a comma in the DN, the comma must be escaped with a backslash (\).
objectClass: top	<i>Required.</i> Specifies the <code>top</code> object class.
objectClass: person	Specifies the <code>person</code> 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.

LDIF Element	Description
objectClass: organizationalPerson	Specifies the <code>organizationalPerson</code> object class. This object class specification should be included because some LDAP clients require it during search operations for an organizational person.
objectClass: inetOrgPerson	Specifies the <code>inetOrgPerson</code> object class. The <code>inetOrgPerson</code> object class is recommended for the creation of an organizational person entry because this object class includes the widest range of attributes. The <code>uid</code> attribute is required by this object class, and entries that contain this object class are named based on the value of the <code>uid</code> attribute.
cn: <i>common_name</i>	Specifies the person's common name, which is the full name commonly used by the person. For example, cn: Bill Anderson. At least one common name is required.
sn: <i>surname</i>	Specifies the person's surname, or last name. For example, sn: Anderson. A surname is required.
<i>list_of_attributes</i>	Specifies the list of optional attributes to maintain for the entry.

Table A.4. LDIF Elements in Person Entries

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, do the following:

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 blank, or else the `ldapmodify` utility will exit. For more information, refer to [Section 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 *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:

- *Initializing the database through the Directory Server Console.* Use this method if there is a small database to import (less than 10,000 entries). See [Section 1.1, "Importing a Database from the 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 1.3.1, "Importing Using the ldif2db Command-Line Script"](#).
- `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 2.4, “Adding and Modifying Entries Using `ldapmodify`”](#).

5.1. LDIF File Example

The following example shows an LDIF file that 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
```

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 2, “Identifying 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:: Aas1jdoaAJASI023909jaASJaonasd0ADS
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 B. Finding Directory Entries

Entries in the directory can be searched for and found using any LDAP client. Most clients provide some form of search interface so that the directory can be searched easily and entry information can be easily retrieved.



NOTE

Users cannot search the directory unless the appropriate access control has been set in the directory. For information on setting access control in the directory, see [Chapter 6, Managing Access Control](#).

1. Finding Entries Using the Directory Server Console

Users can browse the **Directory** tab of the Directory Server Console to see the contents of the directory tree and search for specific entries in the directory.

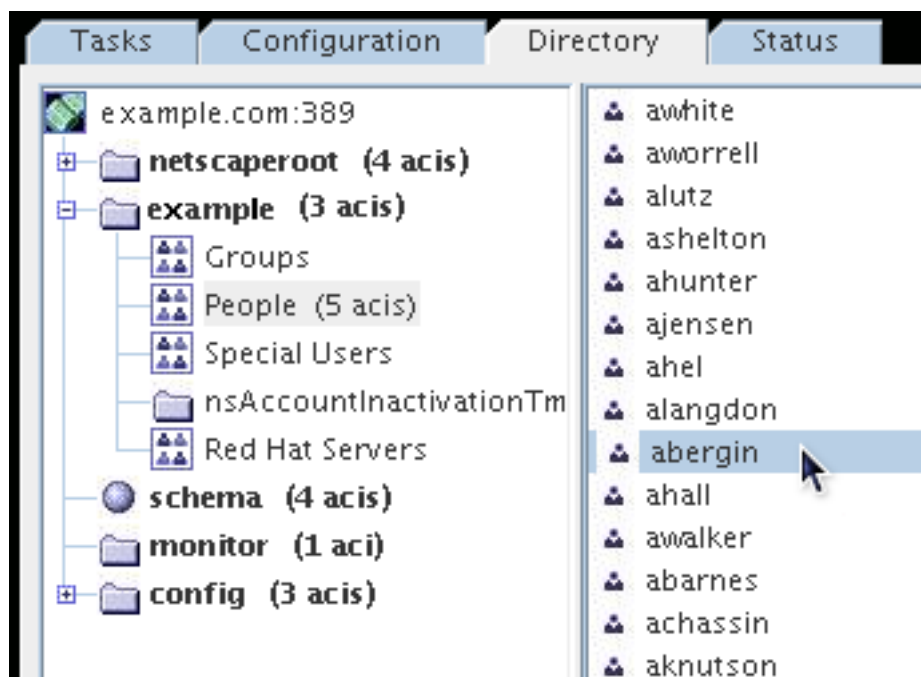


Figure B.1. Browsing Entries in the Directory Tab

Depending on the DN used to authenticate to the directory, this tab displays the contents of the directory that the user account has access permissions to view. Browse through the contents of

the tree, or right-click an entry, and select **Search** from the pop-up menu.

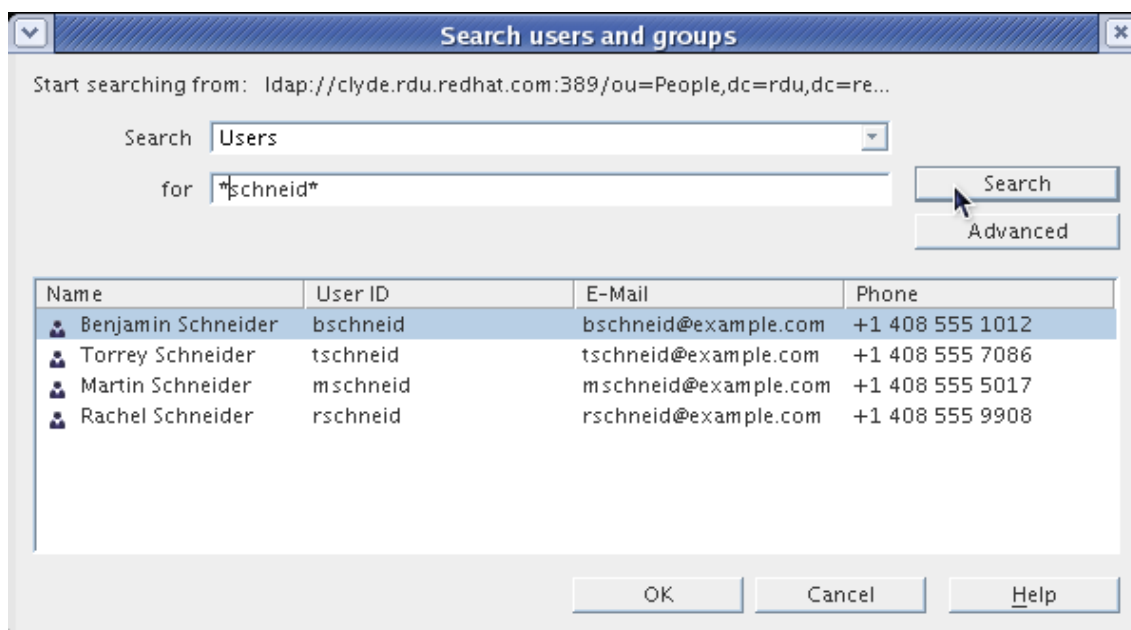


Figure B.2. Searching for Entries



TIP

See the online help for information on using the search form.



CAUTION

Do not modify the contents of the `o=NetscapeRoot` suffix using the **Directory** tab unless instructed to do so by Red Hat technical support.

2. 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 distinguished name and password and locates entries based on a specified search filter. The search scope can include a single entry, an entry's immediate subentries, or an entire tree or subtree.

Search results are returned in LDIF format.

Red Hat Directory Server uses Mozilla LDAP tools, including `ldapsearch`. The MozLDAP tools are installed with Directory Server and are located in the `/usr/lib/mozldap` directory for Red

Hat Enterprise Linux i386, in the `/usr/lib64/mozldap` directory on 64-bit versions of Red Hat Enterprise Linux and Solaris, and in the `/opt/dirsrv/bin/mozldap/` directory on HP-UX. When running any LDAP command, make sure to use the MozLDAP utilities, otherwise the command will return errors.



NOTE

For most Linux systems, OpenLDAP tools are already installed in the `/usr/bin/` directory. These OpenLDAP tools will not work for Directory Server operations.

This section contains information about the following topics:

- [Section 2.1, “Using Special Characters”](#)
- [Section 2.2, “Idapsearch Command-Line Format”](#)
- [Section 2.3, “Commonly Used Idapsearch Options”](#)
- [Section 2.4, “Idapsearch Examples”](#)

2.1. Using Special Characters

When using the `ldapsearch` command-line utility, it may be necessary to specify values that contain characters that have special meaning to the command-line interpreter, such as space (), asterisk (*), or backslash (\). Enclose the value which has the special character in quotation marks ("). For example:

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

Depending on the command-line interpreter, use either single or double quotation marks. In general, use single quotation marks (') to enclose values. Use double quotation marks (") to allow variable interpolation if there are shell variables. Refer to the operating system documentation for more information.

2.2. Idapsearch Command-Line Format

The `ldapsearch` command must use the following format:

```
ldapsearch [optional_options] [optional_search_filter]  
[optional_list_of_attributes]
```

- *optional_options* is a series of command-line options. These must be specified before the

search filter, if any are used.

- *optional_search_filter* is an LDAP search filter as described in [Section 3, “LDAP Search Filters”](#). Do not specify a separate search filter if search filters are specified in a file using the `-f` option.
- *optional_list_of_attributes* is a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see [Section 2.4.6, “Displaying Subsets of Attributes”](#). If a list of attributes is not specified, the search returns values for all attributes permitted by the access control set in the directory (with the exception of operational attributes).



NOTE

For operational attributes to be returned as a result of a search operation, they must be explicitly specified in the search command. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (*) in the list of attributes in the `ldapsearch` command. To retrieve no attributes, just a list of the matching DNs, use the special attribute `1.1`. This is useful, for example, to get a list of DNs to pass to the `ldapdelete` command.

2.3. 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 "ou=groups, dc=example,dc=com"`.

Option	Description
-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 <code>LDAP_BASEDN</code> 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> <div><pre>-b "cn=Barbara Jensen, ou=Product Development, dc=example, dc=com"</pre></div> <p>To search the root DSE entry, specify an empty string here, such as <code>-b ""</code>.</p>
-D	<p>Specifies the distinguished name with which to authenticate to the server. This is optional if</p>

Option	Description
	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, <code>-D "uid=bjensen,dc=example,dc=com"</code> .
<code>-h</code>	Specifies the hostname or IP address of the machine on which the Directory Server is installed. For example, <code>-h mozilla</code> . If a host is not specified, <code>ldapsearch</code> uses the <code>localhost</code> .
<code>-l</code>	Specifies the maximum number of seconds to wait for a search request to complete. For example, <code>-l 300</code> . The default value for the <code>nsslapd-timelimit</code> attribute is 3600 seconds. Regardless of the value specified, <code>ldapsearch</code> will never wait longer than is allowed by the server's <code>nsslapd-timelimit</code> attribute.
<code>-p</code>	Specifies the TCP port number that the Directory Server uses. For example, <code>-p 1049</code> . The default is 389. If <code>-z</code> is used, the default is 636.
<code>-s</code>	Specifies the scope of the search. The scope can be one of the following: <code>base</code> searches only the entry specified in the <code>-b</code> option or defined by the <code>LDAP_BASEDN</code> environment variable. <code>one</code> searches only the immediate children of the entry specified in the <code>-b</code> option. Only the children are searched; the actual entry specified in the <code>-b</code> option is not searched. <code>sub</code> searches the entry specified in the <code>-b</code> option and all of its descendants; that is, perform a subtree search starting at the point identified in the <code>-b</code> option. This is the default.
<code>-w</code>	Gives the password associated with the distinguished name that is specified in the <code>-D</code> option. If this option is not specified, anonymous access is used. For example, <code>-w diner892</code> .
<code>-x</code>	Specifies that the search results are sorted on

Option	Description
	the server rather than on the client. This is useful for sorting according to a matching rule, as with an international search. In general, it is faster to sort on the server rather than on the client.
-z	Sets the maximum number of entries to return in response to a search request. For example, <code>-z 1000</code> . Normally, regardless of the value specified here, <code>ldapsearch</code> never returns more entries than the number allowed by the server's <code>nsslapd-sizelimit</code> attribute. However, this limitation can be overridden by binding as the root DN when using this command-line argument. When binding as the root DN, this option defaults to zero (0). The default value for the <code>nsslapd-sizelimit</code> attribute is 2000 entries.

For detailed information on all `ldapsearch` utility options, refer to the *Directory Server Configuration, Command, and File Reference*.

2.4. ldapsearch Examples

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 4.2, “Defining User Access - userdn Keyword”](#).
- The server is located on a host named `mozilla`.
- The server uses port number 389. Since this is the default port, the port number does not have to be sent in the search request.
- SSL is enabled for the server on port 636(the default SSL port number).
- The suffix under which all data is stored is `dc=example,dc=com`.

2.4.1. 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 -h mozilla -b "dc=example,dc=com" -s sub "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.

2.4.2. 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 -h mozilla -b "dc=example,dc=com" "cn=babs jensen"
```

2.4.3. Searching the Root DSE Entry

The root DSE is a special entry that contains a list of all the suffixes supported by the local Directory Server. This entry can be searched by supplying a search base of "", a search scope of *base*, and a filter of "objectclass=*". For example:

```
ldapsearch -h mozilla -b "" -s base "objectclass=*
```

2.4.4. Searching the Schema Entry

Directory Server stores all directory server schema in the special *cn=schema* entry. This entry contains information on every object class and attribute defined for the Directory Server. The following command searches the contents of the *cn=schema* entry:

```
ldapsearch -h mozilla -b "cn=schema" -s base "objectclass=*
```

2.4.5. 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, suppose *LDAP_BASEDN* is set to *dc=example,dc=com*. Then to search for *cn=babs*

jensen in the directory, use the following command-line call:

```
ldapsearch -h mozilla "cn=babs jensen"
```

In this example, the default scope of `sub` is used because the `-s` option was not used to specify the scope.

2.4.6. Displaying Subsets of Attributes

The `ldapsearch` command returns all search results in LDIF format. By default, `ldapsearch` returns the entry's distinguished name and all of the attributes that a user is allowed to read. The directory access control can be set such that users are allowed to read only a subset of the attributes on any given directory entry. Only operational attributes are not returned. For operational attributes to be returned as a result of a search operation, explicitly specify them in the search command.

It may not be necessary to have all of the attributes for an entry returned in the search results. The returned attributes can be limited to just a few specific attributes by specifying the desired ones on the command line immediately after the search filter. For example, to show the `cn` and `sn` attributes for every entry in the directory, use the following command-line call:

```
ldapsearch -h mozilla "objectclass=*" sn cn
```

This example assumes the search base is set with `LDAP_BASEDN`.

2.4.7. Specifying Search Filters Using a File

Search filters can be entered into a file instead of entering them on the command-line. In this case, specify each search filter on a separate line in the file. The `ldapsearch` command runs each search in the order in which it appears in the file.

For example:

```
sn=Francis
givenname=Richard
```

`ldapsearch` first finds all the entries with the surname `Francis`, then all the entries with the givenname `Richard`. If an entry is found that matches both search criteria, then the entry is returned twice.

For example, suppose the previous search filters were specified in a file named `searchdb`, and the search base is set using `LDAP_BASEDN`. Then the following returns all the entries that match either search filter:

```
ldapsearch -h mozilla -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 -h mozilla -f searchdb sn givenname
```

2.4.8. 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 -h mozilla -s base -b "l=Bolivia\,S.A.,dc=example,dc=com"
"objectclass=*"
```

2.4.9. Using Client Authentication When Searching

This example shows user `bjensen` searching the directory using client authentication:

```
ldapsearch -h mozilla -p 636 -b "dc=example,dc=com" -N "bjensenscertname"
-Z -W certdbpassword -P /home/bjensen/certdb/cert8.db
"givenname=Richard"
```

3. LDAP Search Filters

Search filters select the entries to be returned for a search operation. They are most commonly used with the `ldapsearch` command-line utility. When using `ldapsearch`, there can be multiple search filters in a file, with each filter on a separate line in the file, or a search filter can be specified directly on the command-line.

For example, the following filter specifies a search for the common name Babs Jensen:

```
cn=babs jensen
```

This search filter returns all entries that contain the common name Babs Jensen. Searches for common name values are not case sensitive.

When the common name attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match this filter:

```
cn: babs jensen
cn;lang-fr: babs jensen
```

For a list of all the supported language tags, see [Table D.1, "Supported Locales"](#).

3.1. Search Filter Syntax

The basic syntax of a search filter is:

```
attribute operator value
```

For example:

```
buildingname>=alpha
```

In this example, *buildingname* is the attribute, *>=* is the operator, and *alpha* is the value. Filters can also be defined that use different attributes combined together with Boolean operators.

Search filters are described in detail in the following sections:

- [Section 3.1.1, “Using Attributes in Search Filters”](#)
- [Section 3.1.2, “Using Operators in Search Filters”](#)
- [Section 3.1.3, “Using Compound Search Filters”](#)
- [Section 3.1.4, “Search Filter Examples”](#)

3.1.1. Using Attributes in Search Filters

When searching for an entry, the attributes associated with that type of entry can be specified, such as using the *cn* attribute to search for people with a specific common name.

Examples of attributes that people entries include are the following:

- *cn* for the person's common name.
- *sn* for the person's surname, last name, or family name.
- *telephoneNumber* for the person's telephone number.
- *buildingName* for the name of the building in which the person resides.
- *l* for the physical location of the person.

3.1.2. Using Operators in Search Filters

The operators that can be used in search filters are listed in [Table B.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 4, “Searching an Internationalized Directory”](#).

Search Type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, <code>cn=Bob Johnson</code>
Substring	<code>=string* string</code>	Returns entries containing attributes containing the specified substring. For example, <code>cn=Bob*</code> <code>cn=*Johnson</code> <code>cn=*John*</code> <code>cn=B*John</code> . 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, <code>buildingname >= alpha</code> .
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, <code>buildingname <= alpha</code> .
Presence	=*	Returns entries containing one or more values for the specified attribute. For example, <code>cn=*</code> <code>telephonenumber=*</code> <code>manager=*</code> .
Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example, <code>cn~=suret l~=san francisco</code> could return <code>cn=sarette l=san francisco</code> .

Table B.1. Search Filter Operators

3.1.3. 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 B.2, “Search Filter Boolean Operators”](#).

Boolean operators can be combined and nested together to form complex expressions, such as:

```
(Boolean-operator(filter)((Boolean-operator(filter)(filter))) )
```

The Boolean operators available for use with search filters include the following:

Operator	Symbol	Description
AND	&	All specified filters must be true for the statement to be true. For example, <i>(&(filter)(filter)(filter)...) </i> .
OR		At least one specified filter must be true for the statement to be true. For example, <i>((filter)(filter)(filter)...) </i> .
NOT	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the NOT operator. For example, <i>(!(filter))</i> .

Table B.2. Search Filter Boolean Operators

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.

3.1.4. Search Filter Examples

The following filter searches for entries containing one or more values for the manager attribute. This is also known as a presence search:

```
manager=*
```

The following filter searches for entries containing the common name `Ray Kultgen`. This is also known as an equality search:

```
cn=Ray Kultgen
```

The following filter returns all entries that do not contain the common name `Ray Kultgen`:

```
(!(cn=Ray Kultgen))
```

The following filter returns all entries that contain a description attribute that contains the substring `x.500`:

```
description=*X.500*
```

The following filter returns all entries whose organizational unit is `Marketing` and whose description field does not contain the substring `x.500`:

```
(&(ou=Marketing)(!(description=*X.500*)))
```

The following filter returns all entries whose organizational unit is `Marketing` and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(|(manager=cn=Julie Fulmer,ou=Marketing,dc=example,dc=com)  
  (manager=cn=Cindy Zwaska,ou=Marketing,dc=example,dc=com)))
```

The following filter returns all entries that do not represent a person:

```
(!(objectClass=person))
```

The following filter returns all entries that do not represent a person and whose common name is similar to `printer3b`:

```
(&(!(objectClass=person))(cn~=printer3b))
```

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 2, “Identifying Supported Locales”](#).



NOTE

An LDAPv3 search is required to perform internationalized searches. Therefore, do not specify the `-v2` option on the call for `ldapsearch`.

This section focuses on the matching rule filter portion of the `ldapsearch` syntax. For more information on general `ldapsearch` syntax, see [Section 3, “LDAP Search Filters”](#). For information on searching internationalized directories using the **Users and Groups** portion of the Red Hat Console, see the online help.

This section covers the following topics:

- [Section 4.1, “Matching Rule Filter Syntax”](#)
- [Section 4.2, “Supported Search Types”](#)
- [Section 4.3, “International Search Examples”](#)

4.1. Matching Rule Filter Syntax

A matching rule provides special guidelines for how the directory compares strings during a search operation. In an international search, the matching rule tells the system what collation order and operator to use when performing the search operation. For example, a matching rule in an international search might tell the server to search for attribute values that come at or after `llama` in the Spanish collation order. The syntax of the matching rule filter is as follows:

```
attr:matchingRule:=value
```

- `attr` is an attribute belonging to entries being searched, such as `cn` or `mail`.
- `matchingRule` is a string that identifies either the collation order or the collation order and a relational operator, depending on the preferred format. For a discussion of matching rule formats, see [Section 4.1.1, “Matching Rule Formats”](#).
- `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 portion of the filter depends on the matching rule format used.

4.1.1. Matching Rule Formats

The matching rule portion of a search filter 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 4.1.1.1, “Using an OID for the Matching Rule”](#)
- [Section 4.1.1.2, “Using a Language Tag for the Matching Rule”](#)
- [Section 4.1.1.3, “Using an OID and Suffix for the Matching Rule”](#)
- [Section 4.1.1.4, “Using a Language Tag and Suffix for the Matching Rule”](#)

4.1.1.1. Using an OID for the Matching Rule

Each locale supported by the Directory Server has an associated collation order OID. For a list of locales supported by the directory server and their associated OIDs, see [Table D.1, “Supported Locales”](#).

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
```

4.1.1.2. Using a Language Tag for the Matching Rule

Each locale supported by the Directory Server has an associated language tag. For a list of locales supported by the directory server and their associated language tags, see [Table D.1, “Supported Locales”](#).

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
```

4.1.1.3. 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
```

For example, to search for *businessCategory* attributes with the value `softwareprodukte` in the German collation order, use the following filter:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
```

The `.3` in the previous example is the equality suffix.

For a list of locales supported by the Directory Server and their associated OIDs, see [Table D.1, “Supported Locales”](#). For a list of relational operators and their equivalent suffixes, see [Table B.3, “Search Types, Operators, and Suffixes”](#).

4.1.1.4. 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
```

For example, to search for all surnames that come at or after `La Salle` in the French collation order, use the following filter:

```
sn:fr.4:=La Salle
```

For a list of locales supported by the Directory Server and their associated language tags, see [Table D.1, “Supported Locales”](#). For a list of relational operators and their equivalent suffixes, refer to [Table B.3, “Search Types, Operators, and Suffixes”](#).

4.1.2. Using Wildcards in Matching Rule Filters

When performing a substring search using a matching rule filter, use the asterisk (*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter `l` and ends with the letter `n`, enter a `l*n` in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter `u`, enter a value of `u*` in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, the asterisk must be escaped with the designated escape sequence, `\5c2a`. For example, to search for all employees with `businessCategory` attribute values of `Example*Net product line`, enter the following value in the search filter:

```
Example\5c2a*Net product line
```

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 B.3, “Search Types, Operators, and Suffixes”](#) summarizes each type of search, the operator, and the equivalent suffix.

Search Type	Operator	Suffix
Less-than	<	.1
Less-than or equal-to	<=	.2
Equality	=	.3
Greater-than or equal-to	>=	.4
Greater-than	>	.5

Search Type	Operator	Suffix
Substring	*	.6

Table B.3. Search Types, Operators, and Suffixes

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.

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
```

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
```

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
```

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
```

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 `schranka4` in the Czechoslovakian 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:=schranka4
...
mailHost:cs.5:=schranka4
```

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)
```

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 such as the Directory Server Gateway.
- 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>.

1. Components of an LDAP URL

LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base_dn?attributes?scope?filter
```

The `ldap://` protocol is used to connect to LDAP servers over unsecured connections, and the `ldaps://` protocol is used to connect to LDAP servers over TLS/SSL 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>.

Component	Description
hostname	Name (or IP address in dotted format) of the LDAP server. For example, <code>ldap.example.com</code> or <code>192.202.185.90</code> .
port	Port number of the LDAP server (for example, 696). If no port is specified, the standard LDAP port (389) or LDAPS port (636) is used.

Component	Description
base_dn	Distinguished name (DN) of an entry in the directory. This DN identifies the entry that is the starting point of the search. If no base DN is specified, the search starts at the root of the directory tree.
attributes	The attributes to be returned. To specify more than one attribute, use commas to separate the attributes; for example, <code>cn,mail,telephoneNumber</code> . If no attributes are specified in the URL, all attributes are returned.
scope	<p>The scope of the search, which can be one of these values:</p> <p><code>base</code> retrieves information only about the distinguished name (<i>base_dn</i>) specified in the URL.</p> <p><code>one</code> 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><code>sub</code> 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 <code>base</code> 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 <code>(objectClass=*)</code> .

Table C.1. LDAP URL Components

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.

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
"	%22
#	%23
%	%25
{	%7b
}	%7d
	%7c
\	%5c
^	%5e
~	%7e
[%5b
]	%5d
`	%60

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. For example, Directory Server Gateway supports authentication.

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.

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 Czechoslovakian and `10 janvier 1996` in French.

Because a locale describes cultural, customary, and regional differences in addition to mechanical language differences, the directory data can both be translated into the specific languages understood by users as well as be presented in a way that users in a given region expect.

2. Identifying 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 10, Managing Indexes](#).

[Table D.1, “Supported Locales”](#) lists each locale supported by Directory Server and identifies the associated language tags and OIDs.

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
Albanian	sq	2.16.840.1.113730.3.3.2.44.1
Arabic	ar	2.16.840.1.113730.3.3.2.1.1
Belorussian	be	2.16.840.1.113730.3.3.2.2.1
Bulgarian	bg	2.16.840.1.113730.3.3.2.3.1
Catalan	ca	2.16.840.1.113730.3.3.2.4.1

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
Chinese (Simplified)	zh	2.16.840.1.113730.3.3.2.49.1
Chinese (Traditional)	zh-TW	2.16.840.1.113730.3.3.2.50.1
Croatian	hr	2.16.840.1.113730.3.3.2.22.1
Czechoslovakian	cs	2.16.840.1.113730.3.3.2.5.1
Danish	da	2.16.840.1.113730.3.3.2.6.1
English (US)	en or en-US	2.16.840.1.113730.3.3.2.11.1
Estonian	et	2.16.840.1.113730.3.3.2.16.1
Finnish	fi	2.16.840.1.113730.3.3.2.17.1
French	fr or fr-FR	2.16.840.1.113730.3.3.2.18.1
German	de	2.16.840.1.113730.3.3.2.7.1
Greek	el	2.16.840.1.113730.3.3.2.10.1
Hebrew	iw	2.16.840.1.113730.3.3.2.27.1
Hungarian	hu	2.16.840.1.113730.3.3.2.23.1
Icelandic	is	2.16.840.1.113730.3.3.2.24.1
Japanese	ja	2.16.840.1.113730.3.3.2.28.1
Korean	ko	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish	lv	2.16.840.1.113730.3.3.2.31.1
Lithuanian	lt	2.16.840.1.113730.3.3.2.30.1
Macedonian	mk	2.16.840.1.113730.3.3.2.32.1
Norwegian	no	2.16.840.1.113730.3.3.2.35.1
Polish	pl	2.16.840.1.113730.3.3.2.38.1
Romanian	ro	2.16.840.1.113730.3.3.2.39.1
Russian	ru	2.16.840.1.113730.3.3.2.40.1
Serbian (Cyrillic)	sr	2.16.840.1.113730.3.3.2.45.1
Serbian (Latin)	sh	2.16.840.1.113730.3.3.2.41.1
Slovakian	sk	2.16.840.1.113730.3.3.2.42.1
Slovenian	sl	2.16.840.1.113730.3.3.2.43.1
Spanish	es or es-ES	2.16.840.1.113730.3.3.2.15.1
Swedish	sv	2.16.840.1.113730.3.3.2.46.1
Turkish	tr	2.16.840.1.113730.3.3.2.47.1
Ukrainian	uk	2.16.840.1.113730.3.3.2.48.1

Table D.1. Supported Locales

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 1.3.8, “Adding an Attribute Subtype”](#). [Table D.2, “Supported Language Subtypes”](#) lists the supported language subtypes for Directory Server.

Language Tag	Language
af	Afrikaans
be	Belorussian
bg	Bulgarian
ca	Catalan
cs	Czechoslovakian
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fi	Finnish
fo	Faroese
fr	French
ga	Irish
gl	Galician
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ko	Korean
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sk	Slovakian

Language Tag	Language
sl	Slovenian
sq	Albanian
sr	Serbian
sv	Swedish
tr	Turkish
uk	Ukrainian
zh	Chinese

Table D.2. Supported Language Subtypes

4. Troubleshooting Matching Rules

International collation order matching rules may not behave consistently. Some forms of matching-rule invocation do not work correctly, producing incorrect search results. For example, the following rules do not work:

```
ldapsearch -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w password
-b "dc=example,dc=com" "sn:2.16.840.1.113730.3.3.2.7.1:=passin"

ldapsearch -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w password
-b "dc=example,dc=com" "sn:de:=passin"
```

However, the rules listed below will work (note the .3 before the `passin` value):

```
ldapsearch -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w password
-b "dc=example,dc=com" "sn:2.16.840.1.113730.3.3.2.7.1.3:=passin"

ldapsearch -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w password
-b "dc=example,dc=com" "sn:de.3:=passin"
```

Glossary

A

access control instruction	See ACI .
ACI	An instruction that grants or denies permissions to entries in the directory. See Also access control instruction .
access control list	See ACL .
ACL	The mechanism for controlling access to your directory. See Also access control list .
access rights	In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.
account inactivation	Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.
All IDs Threshold	<i>Replaced with the ID list scan limit in Directory Server version 7.1.</i> A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token. See Also ID list scan limit .
All IDs token	A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.
anonymous access	When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.
approximate index	Allows for efficient approximate or "sounds-like" searches.
attribute	Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute

	value.
attribute list	A list of required and optional attributes for a given entry type or object class.
authenticating directory server	In pass-through authentication (PTA), the authenticating Directory Server is the Directory Server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.
authentication	<p>(1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.</p> <p>(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.</p>
authentication certificate	Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

B

base DN	Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.
base distinguished name	See base DN .
bind DN	Distinguished name used to authenticate to Directory Server when performing an operation.
bind distinguished name	See bind DN .
bind rule	In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.
branch entry	An entry that represents the top of a subtree in the directory.
browser	Software, such as Mozilla Firefox, used to request and view World Wide Web material stored as HTML files. The browser

	uses the HTTP protocol to communicate with the host server.
browsing index	Speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branch point in the directory tree to improve display performance. See Also virtual list view index .

C

CA	See Certificate Authority .
cascading replication	In a cascading replication scenario, one server, often called the hub supplier, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. It receives updates from the supplier server that holds the master copy of the data and in turn supplies those updates to the consumer.
certificate	A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in the directory as user object attributes.
Certificate Authority	Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a CA .
CGI	Common Gateway Interface. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts and can be written in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.
chaining	A method for relaying requests to another server. Results for the request are collected, compiled, and then returned to the client.
changelog	A changelog is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on replica servers or on other masters, in the case of multi-master replication.
character type	Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext	Encrypted information that cannot be read by anyone without the proper key to decrypt the information.
class definition	Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.
class of service	See CoS .
classic CoS	A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.
client	See LDAP client .
code page	An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.
collation order	Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.
consumer	Server containing replicated directory trees or subtrees from a supplier server.
consumer server	In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.
CoS	A method for sharing attributes between entries in a way that is invisible to applications.
CoS definition entry	Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.
CoS template entry	Contains a list of the shared attribute values. See Also template entry .

D

daemon	A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.
DAP	Directory Access Protocol. The ISO X.500 standard protocol that provides client access to the directory.

data master	The server that is the master source of a particular piece of data.
database link	An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.
default index	One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.
definition entry	See CoS definition entry .
Directory Access Protocol	See DAP .
directory tree	The logical representation of the information stored in the directory. It mirrors the tree model used by most filesystems, with the tree's root point appearing at the top of the hierarchy. Also known as DIT .
Directory Manager	The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the Directory Manager.
Directory Server Gateway	A collection of CGI forms that allows a browser to perform LDAP client functions, such as querying and accessing a Directory Server, from a web browser. Also called DSGW .
directory service	A database application designed to manage descriptive, attribute-based information about people and resources within an organization.
distinguished name	String representation of an entry's name and location in an LDAP directory.
DIT	See directory tree .
DN	See distinguished name .
DM	See Directory Manager .
DNS	Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as <code>www.example.com</code>). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.
DNS alias	A DNS alias is a hostname that the DNS server knows points

to a different host#specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as `www.yourdomain.domain` might point to a real machine called `realthing.yourdomain.domain` where the server currently exists.

DSGW

See [Directory Server Gateway](#).

E

entry A group of lines in the LDIF file that contains information about an object.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

F

file extension The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename `index.html` the file extension is `html`.

file type The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter A constraint applied to a directory query that restricts the information returned.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

G

gateway	See Directory Server Gateway .
general access	When granted, indicates that all authenticated users can access directory information.
GSS-API	Generic Security Services. The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

H

hostname	A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, <code>www.example.com</code> is the machine <code>www</code> in the subdomain <code>example</code> and <code>com</code> domain.
HTML	Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Mozilla Firefox how to display text, position graphics, and form items and to display links to other pages.
HTTP	Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.
HTTPD	An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an <code>httpd</code> .
HTTPS	A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.
hub	In the context of replication, a server that holds a replica that is copied from a different server, and, in turn, replicates it to a third server. See Also cascading replication .

I

ID list scan limit	A size limit which is globally applied to any indexed search operation. When the size of an individual ID list reaches this limit, the server replaces that ID list with an all IDs token.
index key	Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS	An indirect CoS identifies the template entry using the value of one of the target entry's attributes.
international index	Speeds up searches for information in international directories.
International Standards Organization IP address	See ISO . <i>Also Internet Protocol address.</i> A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10).
ISO	International Standards Organization.

K

knowledge reference	Pointers to directory information stored in different databases.
---------------------	--

L

LDAP	Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.
LDAPv3	Version 3 of the LDAP protocol, upon which Directory Server bases its schema format.
LDAP client	Software used to request and view LDAP entries from an LDAP Directory Server. See Also browser .
LDAP Data Interchange Format LDAP URL	See LDAP Data Interchange Format . Provides the means of locating Directory Servers using DNS and then completing the query via LDAP. A sample LDAP URL is <code>ldap://ldap.example.com</code> .
LDBM database	A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.
LDIF	LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.
leaf entry	An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Access Protocol

See [LDAP](#).

locale

Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

M

managed object

A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role

Allows creation of an explicit enumerated list of members.

management information
base
mapping tree

See [MIB](#).

A data structure that associates the names of suffixes (subtrees) with databases.

master

See [supplier](#).

master agent

See [SNMP master agent](#).

matching rule

Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5

A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data that is unique with high probability and is mathematically extremely hard to produce; a piece of data that will produce the same message digest.

MD5 signature

A message digest produced by the MD5 algorithm.

MIB

Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree-like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace

Management Information Base namespace. The means for directory data to be named and referenced. Also called the

directory tree.

monetary format	Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.
multi-master replication	An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a changelog for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.
multiplexor	The server containing the database link that communicates with the remote server.

N

n + 1 directory problem	The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.
name collisions	Multiple entries with the same distinguished name.
nested role	Allows the creation of roles that contain other roles.
network management application	Network Management Station component that graphically displays information about SNMP managed devices, such as which device is up or down and which and how many error messages were received.
network management station NIS	See NMS . Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, filesystems, and network parameters throughout a network of computers.
NMS	Powerful workstation with one or more network management applications installed. Also network management station .
ns-slapd	Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server. See Also slapd .

O

object class	Defines an entry type in the directory by defining which attributes are contained in the entry.
object identifier	A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations. See Also OID .
OID	See object identifier .
operational attribute	Contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

P

parent access	When granted, indicates that users have access to entries below their own in the directory tree if the bind DN is the parent of the targeted entry.
pass-through authentication	See PTA .
pass-through subtree	In pass-through authentication, the PTA directory server will pass through bind requests to the authenticating directory server from all clients whose DN is contained in this subtree.
password file	A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as <code>/etc/passwd</code> because of where it is kept.
password policy	A set of rules that governs how passwords are used in a given directory.
permission	In the context of access control, permission states whether access to the directory information is granted or denied and the level of access that is granted or denied. See Also access rights .
PDU	Encoded messages which form the basis of data exchanges between SNMP devices. Also protocol data unit .
pointer CoS	A pointer CoS identifies the template entry using the template DN only.
presence index	Allows searches for entries that contain a specific indexed attribute.

protocol	A set of rules that describes how devices on a network exchange information.
protocol data unit	See PDU .
proxy authentication	A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.
proxy DN	Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.
PTA	Mechanism by which one Directory Server consults another to check bind credentials. Also pass-through authentication .
PTA directory server	In pass-through authentication (PTA), the PTA Directory Server is the server that sends (passes through) bind requests it receives to the authenticating directory server .
PTA LDAP URL	In pass-through authentication, the URL that defines the authenticating directory server , pass-through subtree(s), and optional parameters.

R

RAM	Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.
rc.local	A file on Unix machines that describes programs that are run when the machine starts. It is also called <code>/etc/rc.local</code> because of its location.
RDN	The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name. Also relative distinguished name .
referential integrity	Mechanism that ensures that relationships between related entries are maintained within the directory.
referral	<p>(1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.</p> <p>(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding</p>

	process is called a referral.
read-only replica	A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.
read-write replica	A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.
relative distinguished name	See RDN .
replica	A database that participates in replication.
replica-initiated replication	Replication configuration where replica servers, either hub or consumer servers, pull directory data from supplier servers. This method is available only for legacy replication.
replication	Act of copying directory trees or subtrees from supplier servers to replica servers.
replication agreement	Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the replica servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.
RFC	Request for Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.
role	An entry grouping mechanism. Each role has <i>members</i> , which are the entries that possess the role.
role-based attributes	Attributes that appear on an entry because it possesses a particular role within an associated CoS template.
root	The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.
root suffix	The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

S

SASL	An authentication framework for clients as they attempt to bind to a directory. Also Simple Authentication and Security Layer .
------	---

schema	Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.
schema checking	Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default, and users will receive an error if they try to save an entry that does not conform to the schema.
Secure Sockets Layer	See SSL .
self access	When granted, indicates that users have access to their own entries if the bind DN matches the targeted entry.
Server Console	Java-based application that allows you to perform administrative management of your Directory Server from a GUI.
server daemon	The server daemon is a process that, once running, listens for and accepts requests from clients.
server service	A process on Windows that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.
Server Selector	Interface that allows you select and configure servers using a browser.
service	A background process on a Windows machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.
SIE	Server Instance Entry. The ID assigned to an instance of Directory Server during installation.
Simple Authentication and Security Layer	See SASL .
Simple Network Management Protocol	See SNMP .
single-master replication	The most basic replication scenario in which multiple servers, up to four, each hold a copy of the same read-write replicas to replica servers. In a single-master replication scenario, the supplier server maintains a changelog.
SIR	See supplier-initiated replication .
slapd	LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication.

	See Also ns-slapd .
SNMP	Used to monitor and manage application processes running on the servers by exchanging data about network activity. Also Simple Network Management Protocol .
SNMP master agent	Software that exchanges information between the various subagents and the NMS.
SNMP subagent	Software that gathers information about the managed device and passes the information to the master agent. Also called a subagent .
SSL	A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP. Also called Secure Sockets Layer .
standard index	index maintained by default.
sub suffix	A branch underneath a root suffix.
subagent	See SNMP subagent .
substring index	Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.
suffix	The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.
superuser	The most privileged user available on Unix machines. The superuser has complete access privileges to all files on the machine. Also called root .
supplier	Server containing the master copy of directory trees or subtrees that are replicated to replica servers.
supplier server	In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.
supplier-initiated replication	Replication configuration where supplier servers replicate directory data to any replica servers.
symmetric encryption	Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.
system index	Cannot be deleted or modified as it is essential to Directory Server operations.

T

target	In the context of access control, the target identifies the directory information to which a particular ACI applies.
target entry	The entries within the scope of a CoS.
TCP/IP	Transmission Control Protocol/Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.
template entry	See CoS template entry .
time/date format	Indicates the customary formatting for times and dates in a specific region.
TLS	The new standard for secure socket layers; a public key based protocol. Also Transport Layer Security .
topology	The way a directory tree is divided among physical servers and how these servers link with one another.
Transport Layer Security	See TLS .

U

uid	A unique number associated with each user on a Unix system.
URL	Uniform Resource Locator. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is <i>protocol://machine:port/document</i> . The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

V

virtual list view index	Speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branch point in the directory tree to improve display performance. See Also browsing index .
-------------------------	--

X

X.500 standard

The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation.

Index

A

access control

- ACI attribute, 169
- ACI syntax, 173
- allowing or denying access, 181
- and replication, 241
- and schema checking, 177
- anonymous access, 187
- bind rules, 184
 - access at specific time or day, 198
 - access based on value matching, 191
 - general access, 187
 - user and group access, 186
- Boolean bind rules, 201
- compatibility with earlier versions, 241
- creating from console, 202
- dynamic targets, 187
- from specific domain, 197
- from specific IP address, 196
- logging information, 216
- overview, 169
- permissions, 180
- placement of ACIs, 170
- rights, 181
- roles, 142
- SASL authentication, 200
- simple authentication, 199
- SSL authentication, 200
- structure of ACIs, 169
- target DN
 - containing comma, 175
- target DN containing comma, 234
- targeting, 173
- targeting attribute values, 179
- targeting attributes, 176
- targeting entries, 175
- targeting using filters, 178
- using the Access Control Editor, 202
- value matching, 191
- viewing
 - Access Control Editor, 210
 - get effective rights, 211

Access Control Editor

- displaying, 203

access control instruction (ACI). See ACI, 169

access log

- configuring, 434
- manually rotating, 438
- turning off, 434
- turning on, 434
- viewing, 433

account inactivation, 261

- from command line, 262
- from console, 262

account lockout, 258

- configuration
 - attributes, 258
- configuring, 257
 - using command line, 258
 - using console, 258
- disabling, 258
- enabling, 258
- lockout duration, 258
- password failure counter, 258
- replicating attributes, 331

ACI

- assessment, 169
- attribute, 170
- authmethod keyword, 199
- bind rules, 173
- cascading chaining, 96
- creating from console, 204
- dayofweek keyword, 198
- deleting from console, 210
- dns keyword, 197
- editing from console, 209
- evaluation, 170
- examples of use, 216
- groupdn keyword, 190
- inheritance, 194
- ip keyword, 196
- local evaluation
 - cascading chaining, 97
- name, 173
- permissions, 173
- precedence rule, 170
- proxy rights example, 234
- replication, 241
- rights, 181

- roledn keyword, 190
- structure, 169
- syntax, 173
- targattrfilters keyword, 179
- target, 173
- target DN
 - containing comma, 175
- target DN containing comma, 234
- target keywords, 174
- target overview, 173
- targetattr keyword, 176
- targetfilter keyword, 178
- userattr and parent, 194
- userattr keyword, 191
- using macro ACIs, 235
- value-based, 179
- viewing current, 210
- wildcard in target, 176
- wildcards, 188
- ACI attribute
 - default index for, 367
 - overview, 169
- ACI placement, 170
- ACI targets, 175
- ACL. See ACI, 169
- activating accounts
 - from command line, 263
 - from console, 263
- Active Directory
 - schema differences between Directory Server, 534
- add right, 181
- adding directory entries, 27
- Administration Server
 - starting and stopping, 6
- algorithm
 - metaphone phonetic algorithm, 369
 - search, 367
- All IDs Threshold, 388
- all keyword, 187
- allowed attributes
 - creating, 360
 - deleting, 360
 - editing in object class, 361
- allowing access, 181
- anonymous access, 199
 - example, 189
 - overview, 187
- anyone keyword, 187
- approximate index, 363
 - query string codes, 369
- approximate search, 560
- attribute
 - ACI, 169
 - adding, 36
 - adding multiple values, 21
 - adding to entry, 19
 - creating, 355, 360
 - deleting, 36, 356
 - deleting from object class, 360
 - deleting using LDIF update statements, 39
 - editing, 356
 - multi-valued, 356
 - nsslapd-schemacheck, 362
 - OID, 356
 - passwordChange, 247
 - passwordExp, 247
 - passwordGraceLimit, 247
 - passwordInHistory, 249
 - passwordMaxRepeats, 250
 - passwordMin8bit, 251
 - passwordMinAlphas, 250
 - passwordMinCategories, 250
 - passwordMinDigits, 250
 - passwordMinLowers, 250
 - passwordMinSpecials, 250
 - passwordMinTokenLength, 251
 - passwordMinUppers, 251
 - passwordMustChange, 247
 - passwordStorageScheme, 251
 - ref, 108
 - removing a value, 21
 - roles, 139
 - searching for, 560
 - standard, 353
 - syntax, 356
 - targeting, 176
 - user-defined, 354
 - very large, 20
 - viewing, 353
- attribute subtypes, 22
 - adding, 23
 - binary, 22
 - language, 22

- pronunciation, 23
- attribute type field (LDIF), 540
- attribute uniqueness plug-in
 - creating an instance of, 506
- attribute uniqueness plug-in. See unique
- attribute plug-in, 503
- attribute value field (LDIF), 540
- attribute values
 - adding, 36
 - deleting, 39
 - modifying, 38
 - replacing, 36
 - syntax, 356
- attributes values
 - targeting, 179
- audit log
 - configuring, 437
 - disabling, 437
 - enabling, 437
 - viewing, 437
- authentication
 - access control and, 199
 - bind DN, 8
 - certificate-based, 415
 - LDAP URLs, 575
 - over SSL, 405
 - SASL, 421
 - SASL mechanisms, 421
- authmethod keyword, 199

B

- backing up data, 124
 - all, 124
 - db2bak, 125
 - dse.ldif, 126
- bak2db script, 127
- bak2db.pl perl script, 127
- base 64 encoding, 541
- base DN, ldapsearch and, 557
- binary data, LDIF and, 541
- binary subtype, 22
- bind credentials
 - for database links, 78
- bind DN
 - accessing the server, 8
 - resource limits based on, 264

- viewing current, 9
- bind rules
 - access at specific time or day, 198
 - access based on authentication method, 199
 - LDIF example, 200
 - access based on value matching
 - overview, 191
 - ACI syntax, 173
 - all keyword, 187
 - anonymous access, 187
 - example, 189
 - LDIF example, 189
 - anyone keyword, 187
 - authmethod keyword, 199
 - Boolean, 201
 - dayofweek keyword, 198
 - dns keyword, 197
 - general access, 187
 - example, 189
 - group access, 190
 - group access example, 224
 - groupdn keyword, 190
 - ip keyword, 196
 - LDAP URLs, 187
 - LDIF keywords, 185
 - overview, 184
 - parent keyword, 187
 - role access, 190
 - roledn keyword, 190
 - self keyword, 187
 - timeofday keyword, 198
 - user access
 - LDIF example, 188
 - parent, 187
 - self, 187
 - user access example, 219
 - userattr keyword, 191
 - userdn keyword, 186
- Boolean bind rules
 - example, 201
 - overview, 201
- Boolean operators, in search filters, 562
- browsing index, 364

C

- cascading chaining
 - client ACIs, 97
 - configuration attributes, 98
 - configuring defaults, 94
 - configuring from command line, 96
 - configuring from console, 95
 - example, 99
 - local ACI evaluation, 97
 - loop detection, 98
 - overview, 92
 - proxy admin user ACI, 96
 - proxy authorization, 96
- cascading replication
 - initializing the replicas, 305
 - introduction, 274
 - setting up, 298
- certificate
 - mapping to a DN, 416
 - password, 410
- certificate database
 - password, 393
- certificate-based authentication, 415
 - setting up, 416
- chaining
 - cascading, 92
 - component operations, from command line, 73
 - component operations, from console, 72
 - overview, 69
 - using SSL, 85
- change operations, 33
 - add, 36
 - delete, 36
 - replace, 36
- change type
 - add, 33
 - delete, 40
 - LDIF, 32
 - modify, 36
- changelog, 268
 - deleting, 323
- character type, 577
- checkpoint interval, 467
- ciphers
 - list of
 - SSLv3, 413
 - TLSv1, 413
 - none,MD5
 - MD5 message authentication, 415
 - overview, 412
 - selecting, 412
- class of service (CoS), 143
 - access control, 162
 - classic
 - example, 147
 - overview, 147
 - cosPriority attribute, 157
 - creating, 149
 - definition entry, 153
 - editing, 152
 - indirect
 - example, 146
 - overview, 146
 - pointer
 - example, 145
 - overview, 145
 - qualifiers, 154
 - template entry
 - creating, 151
 - overview, 145
- classic CoS
 - example, 147
 - overview, 147
- client
 - using to find entries, 551
- client authentication
 - over SSL, 416
- code page, 577
- collation order
 - international index, 373
 - overview, 577
 - search filters and, 563
- command line
 - providing input from, 24
- command-line scripts
 - db2bak, 125
- command-line utilities
 - certificate-based authentication and, 415
 - ldapdelete, 29
 - ldapmodify, 26
 - ldapsearch, 559
 - ldif, 542
 - ldif2db, 376
- commas, in DNs, 31, 175

- using ldapsearch with, 559
- compare right, 181
- compatibility
 - ACIs, 241
 - replication, 269
- compound search filters, 561
- configuration attributes
 - account lockout, 258
 - cascading chaining, 98
 - password policy, 247
 - suffix, 52
- connections
 - monitoring, 441
 - viewing number of, 440
- consumer initialization
 - filesystem replica, 327
 - manual consumer creation, 326
 - online consumer creation, 324
- consumer server, 268
- continued lines
 - in LDIF, 540
 - in LDIF update statements, 33
- CoS definition entry
 - attributes, 154
 - object classes, 153
- CoS qualifiers, 154
 - default, 154
 - override, 154
- CoS template entry, 145
 - creating, 151
- CoS. See class of service., 143
- cosPriority attribute, 157
- counter, password failures, 258
- country code, 578
- creating a database
 - from the command line, 59
 - from the console, 58
- creating a virtual DIT, 162
- creating the directory, 546
- custom distribution function
 - adding to suffix, 60
- custom distribution logic
 - adding databases, 59
 - adding to suffix, 60

D

- dash, in change operation, 33
- data consistency
 - using referential integrity, 41
- database
 - and associated suffix, 47
 - backing up
 - db2bak, 125
 - backup, 124
 - backup files, 125
 - backup from console, 124
 - creating from command line, 59
 - creating from console, 58
 - creating multiple, 59
 - creating using LDIF, 546
 - deleting, 63
 - export, 119
 - db2ldif, 122
 - encrypted database, 67
 - export from console, 121
 - import, 113
 - encrypted database, 67
 - ldif2db, 116
 - ldif2db.pl, 118
 - ldif2ldap, 119
 - initialization, 115
 - making read-only, 61
 - monitoring from command line, 448
 - monitoring from server console, 445
 - overview, 56
 - read-only mode, 61
 - replication, 267
 - restore, 124
 - restoring, 466
 - bak2db, 127
 - bak2db.pl, 127
 - restoring from console, 126
 - selecting for monitoring, 445
 - viewing backend information, 445
- database encryption, 64
 - importing and exporting, 67
- database link
 - cascading
 - configuring defaults, 94
 - configuring from command line, 96
 - configuring from console, 95
 - overview, 92
 - chaining with SSL, 85

- configuration, 75
- configuration attributes, 81
- configuration example, 81
- configuring bind credentials, 78
- configuring failover servers, 81
- configuring LDAP URL, 80
- configuring suffix, 77
- creating from command line, 77
- creating from console, 75
- deleting, 86
- maintaining remote server info, 86
- overview, 69
- database server parameters
 - read-only, 446
- database transaction logging
 - described, 466
 - durable transactions, 468
 - log file location, 466
- databases
 - in Directory Server, 47
- date format, 578
- dayofweek keyword, 198
- db2bak script, 125
- db2bak utility, 125
- db2ldif utility, 122
- default CoS qualifier, 154
- default referrals
 - setting, 106
 - setting from console, 106
 - settings from command line, 106
- defining
 - access control policy, 202
 - attributes, 355
 - object classes, 359
- definition entry. See CoS definition entry., 154
- delete right, 181
- deleting
 - ACI, 210
 - attribute values, 39
 - attributes, 36, 356
 - attributes from an object class, 360
 - database link, 86
 - entries, 40
 - multiple attributes, 36
 - object classes, 361
- deleting directory entries, 29
- denying access, 181
- precedence rule, 170
- directory creation, 546
- directory entries
 - adding using LDIF, 26
 - creating, 16
 - deleting, 23
 - managing from command line, 24
 - managing from console, 15
 - modifying, 18
 - moving, 36
 - renaming, 36
- Directory Manager
 - attribute, 12
 - configuring, 12
 - privileges, 12
- Directory Server, 438
 - attributes, 12
 - basic administration, 1
 - binding to, 8
 - changing bind DN, 8
 - configuration, 9
 - configuring SASL authentication at startup, 429
 - controlling access, 169
 - creating a root entry, 15
 - creating content, 113
 - creating entries, 16
 - data, 113
 - databases, 47
 - deleting entries, 23
 - file locations, 1
 - importing data, 113
 - international character sets, 577
 - login, 8
 - managing entries, 15
 - MIB, 457
 - modifying entries, 18
 - monitoring, 431
 - monitoring from command line, 443
 - monitoring with SNMP, 453
 - overview, 1
 - performance counters, 438
 - plug-ins, 471
 - starting and stopping, 6
 - starting the Console, 7
 - suffixes, 47
 - supported languages, 578

Directory Server Console

- starting, 7

directory trees

- finding entries in, 552

- disabling suffixes, 55

disk space

- access log and, 434

- log files and, 438

- distribution function, 59

- dn field (LDIF), 540

- dns keyword, 197

dse.ldif

- PTA plugin, 496

dse.ldif file

- backing up, 126

- PTA syntax, 496

- restoring, 129

- durable transactions, 468

- dynamic groups, 167

- creating, 167

- modifying, 167

E

editing

- attributes, 356

- object classes, 360

encryption

- database, 64

- end of file marker, 24

- entity table, 460

entries

- adding an object class, 19

- adding attributes, 19

- adding using LDIF, 26

- adding using LDIF update statements, 33

- adding very large attributes, 20

- creating, 16

- using LDIF, 542

- deleting, 23

- using ldapdelete, 29

- deleting using LDIF update statements, 40

- distribution, 58

- finding, 552

- managing, 15

- managing from command line, 24

- managing from console, 15

- modifying, 18

- using ldapmodify, 26

- using LDIF update statements, 36

- moving, 36

- order of creation, 25

- order of deletion, 30

- removing an object class, 19

- renaming, 36

- root, 546

- targeting, 175

- entry distribution, 58

- entry ID list, 388

- environment variables

- LDAP_BASEDN, 557

- EOF marker, 24

- equality index, 363

- equality search, 560

- example, 563

- international example, 568

error log

- access control information, 216

- configuring, 436

- manually rotating, 438

- turning off, 436

- turning on, 436

- viewing, 435

example

- cascading chaining, 99

- exporting data, 119

- db2ldif, 122

- encrypted database, 67

- using console, 121

- extending the directory schema, 353

F

failover servers

- for database links, 81

- File locations, 1

files

- access log, 433

- database backup, 125

- EOF marker, 24

- error log, 435

- id2entry.db4, 367

- Filesystem Hierarchy Standard, 1

- filesystem replica initialization, 327

- filtered role
 - creating, 135
 - example, 141
- finding
 - attributes, 560
 - entries, 552
- format, LDIF, 539

G

- general access
 - example, 189
 - overview, 187
- get effective rights, 211
 - return codes, 215
- global password policy, 243
- glue entries, 346
- greater than or equal to search
 - international example, 569
 - overview, 560
- groupdn keyword, 190
 - LDIF examples, 190
- groupdnattr keyword, 191
- groups
 - access control, 186
 - access control example, 224
 - access to directory, 190
 - differences between Directory Server and Active Directory, 534
 - dynamic, 167
 - creating, 167
 - modifying, 167
 - overview, 165
 - static, 165
 - creating, 166
 - modifying, 166
- GSS-API, 421

H

- hub, 268

I

- id field (LDIF), 540
- id2entry.db4 file, 367
- identity mapping
 - default, 424
- importing data, 113

- encrypted database, 67
- from console, 114
- ldif2ldap, 119
- using ldif2db, 116
- using ldif2db.pl, 118
- inactivating accounts, 261
- inactivating roles, 132
- index types, 363
 - approximate index, 363
 - browsing index, 364
 - equality index, 363
 - international index, 364
 - presence index, 363
 - substring index, 364
 - virtual list view index, 364
- indexes
 - creating dynamically, 373
 - dynamic changes to, 373
 - presence, 367
- indexing, 363
 - creating indexes from console, 372
 - system indexes, 367
- indirect CoS
 - example, 146
 - overview, 146
- init scripts
 - configuring SASL authentication, 429
- initializing databases, 115
- initializing replicas
 - cascading replication, 305
 - filesystem replica, 327
- interaction table, 461
- international charactersets, 577
- international index, 364
 - collation order, 373
- international searches, 563
 - equality, 568
 - examples, 568
 - greater than, 569
 - greater than or equal to, 569
 - less than, 568
 - less than or equal to, 568
 - matching rule filter syntax, 564
 - substring, 569
 - using OIDs, 565
- internationalization
 - character type, 577

-
- collation order, 577
 - country code, 578
 - date format, 578
 - language tag, 578
 - locales and, 577
 - location of files, 578
 - matching rule filters, 564
 - modifying entries, 41
 - monetary format, 577
 - object identifiers and, 578
 - of LDIF files, 549
 - search filters and, 563
 - supported locales, 578
 - time format, 578
- ip keyword, 196
- ## J
- jpeg images, 541
- ## K
- Kerberos, 421
- configuring, 426
 - realms, 427
- ## L
- language code
- in LDIF entries, 549
 - list of supported, 578
- language subtype, 22
- language support
- language tag, 578
 - searching and, 563
 - specifying using locales, 578
- language tags
- described, 578
 - in international searches, 565
 - in LDIF update statements, 41
- LDAP clients
- authentication over SSL, 417
 - certificate-based authentication and, 415
 - monitoring database with, 448
 - monitoring server with, 443
 - schema and, 353
 - using to find entries, 551
- LDAP Data Interchange Format, see LDIF, 32
- LDAP search filters
- DNs with commas and, 559
 - in targets, 178
 - example, 232
 - examples, 178
- LDAP URLs
- components of, 571
 - examples, 573
 - for database links, 80
 - in access control, 187
 - security, 575
 - syntax, 571
- ldap-agent, 455
- ldapdelete utility, 26
- deleting entries, 29
 - DNs with commas and, 31
 - example, 30
- ldapmodify utility, 26
- attributes with language tags, 41
 - creating a root entry, 25
 - creating entries, 27
 - DNs with commas and, 31
 - example, 27
 - example of use, 27
 - modifying entries, 26
 - schema checking and, 27
 - vs. ldapdelete, 26
- ldapsearch utility
- base DN and, 557
 - commonly used options, 554
 - DNs with commas and, 553
 - example of use, 556
 - format, 553
 - international searches, 563
 - limiting attributes returned, 558
 - search filters, 559
 - specifying files, 558
 - using, 552
- LDAP_BASEDN environment variable, 557
- LDIF
- access control keywords
 - groupdnattr, 191
 - userattr, 191
 - adding entries, 26
 - binary data, 541
 - change type, 32
 - entry format, 539
 - organization, 542

- organizational person, 545
 - organizational unit, 544
 - example, 548
 - internationalization and, 549
 - line continuation, 540
 - Server Console and, 26
 - specifying entries
 - organization, 543
 - organizational person, 545
 - organizational unit, 544
 - update statements, 32
 - using to create directory, 546
 - LDIF entries
 - binary data in, 541
 - creating, 542
 - organizational person, 545
 - organizational units, 544
 - organizations, 542
 - internationalization and, 549
 - LDIF files
 - continued lines, 540
 - creating directory using, 546
 - creating multiple entries, 26
 - example, 548
 - importing from Server Console, 26
 - internationalization and, 549
 - LDIF format, 539
 - LDIF update statements, 32
 - adding attributes, 37
 - adding entries, 33
 - continued lines, 33
 - deleting attribute values, 39
 - deleting attributes, 39
 - deleting entries, 40
 - modifying attribute values, 38
 - modifying entries, 36
 - syntax, 33
 - ldif utility
 - converting binary data to LDIF, 542
 - ldif2db utility, 116
 - options, 376
 - ldif2db.pl perl script, 118
 - ldif2ldap utility, 119
 - legacy consumer
 - configuration, 335
 - legacy replication plug-in
 - overview, 270
 - less than or equal to search
 - international example, 568
 - syntax, 560
 - less than search
 - international example, 568
 - syntax, 560
 - local password policy, 243
 - locales
 - defined, 577
 - location of files, 578
 - supported, 578
 - locked accounts, 258
 - lockout duration, 258
 - log files, 431
 - access log, 433
 - audit log, 437
 - database transaction, 466
 - deletion policy, 433
 - error log, 435
 - location of, 438
 - manually rotating, 438
 - rotation policy, 431
 - setting file permissions, 432
 - viewing when server is down, 431
 - logging
 - for WinSync, 536
 - login identity
 - changing, 8
 - viewing, 9
 - loop detection
 - cascading chaining, 98
- ## M
- macro ACIs
 - example, 235
 - overview, 235
 - syntax, 237
 - managed device
 - overview, 453
 - managed object, 453
 - managed role
 - creating, 134
 - example, 140
 - manually rotating log files, 438
 - markerObjectClass keyword, 510
 - matchingRule format, 564

- using language tag, 565
- using language tag and suffix, 566
- using OID, 565
- using OID and suffix, 566
- metaphone phonetic algorithm, 369
- MIB
 - Directory Server, 457
 - redhat-directory.mib, 457
 - entity table, 460
 - entries table, 460
 - interaction table, 461
 - operations table, 458
- modifying
 - attribute values, 38
 - entries, 36
 - international entries, 41
- monetary format, 577
- monitoring
 - database from command line, 448
 - database from server console, 445
 - Directory Server, 431
 - from console, 438
 - log files, 431
 - replication status, 339
 - threads, 440
 - with SNMP, 453
- monitoring from console, 438
- moving entries, 36
- multi-master replication
 - introduction, 271
 - preventing monopolization of the consumer, 297
 - setting up, 286
- multiple search filters, 561

N

- naming conflicts
 - in replication, 343
- nested role
 - creating, 136
 - example, 142
- nsds5ReplicaBusyWaitTime, 297
- nsds5ReplicaSessionPauseTime, 297
- nsRole, 132
- nsslapd-db-checkpoint-interval, 467
- nsslapd-db-durable-transactions, 468

- nsslapd-db-logdirectory, 467
- nsslapd-idlistscanlimit, 369
- nsslapd-lookthroughlimit attribute
 - role in searching algorithm, 368
- nsslapd-maxbersize, 20
- nsslapd-schemacheck attribute, 362
- nsslapd-sizelimit attribute
 - role in searching algorithm, 368
- nsslapd-timelimit attribute
 - role in searching algorithm, 368
- nsview, 162
- nsviewfilter, 162

O

- object class
 - adding to an entry, 19
 - creating, 359
 - deleting, 361
 - editing, 360
 - name, 359
 - OID, 359
 - parent object, 359
 - referral, 108
 - removing from an entry, 19
 - roles, 139
 - standard, 353
 - user-defined, 357
 - viewing, 357
- object identifier (OID), 578
 - attribute, 356
 - in matchingRule, 565
 - object class, 359
- objectClass field (LDIF), 540
- OID, See object identifier, 578
- operations table, 458
- operations, defined, 440
- operators
 - Boolean, 562
 - international searches and, 567
 - search filters and, 560
 - suffix, 567
- optional attributes
 - creating, 360
 - deleting, 360
 - editing, 361
 - editing in object class, 361

organization, specifying entries for, 542
organizational person, specifying entries for, 545
organizational unit, specifying entries for, 544
override CoS qualifier, 154

P

parent access, 187
parent keyword, 187
parent object, 359
pass-through authentication (PTA). See PTA
plug-in, 491
password change extended operation, 256
password file
 Administration Server, 411
 SSL certificate, 410
password policy
 account lockout, 258
 attributes, 247
 configuring, 243
 using command line, 247
 using console, 244
 global, 243
 lockout duration, 258
 managing, 243
 password failure counter, 258
 replicating account lockout attributes, 331
 replication, 260
 subtree level, 243
 user level, 243
Password Sync, 535
 installation directory, 522
 installed files, 522
 installing, 521
 modifying, 535
 setting up SSL, 523
 starting and stopping, 535
 uninstalling, 536
passwordChange attribute, 247
passwordExp attribute, 247
passwordGraceLimit attribute, 247
passwordInHistory attribute, 249
passwordMaxRepeats attribute, 250
passwordMin8bit attribute, 251
passwordMinAlphas attribute, 250
passwordMinCategories attribute, 250
passwordMinDigits attribute, 250
passwordMinLowers attribute, 250
passwordMinSpecials attribute, 250
passwordMinTokenLength attribute, 251
passwordMinUppers attribute, 251
passwordMustChange attribute, 247
passwords
 account lockout, 258
 certificate, 410
 changing, 256
 failure counter, 258
 lockout duration, 258
 policy
 differences between Directory Server
 and Active Directory, 534
 setting, 255
 synching with Active Directory, 535
passwordStorageScheme attribute, 251
PDUs, 453
performance counters, 445
 monitoring the server with, 438
performance tuning
 database, 464
 server, 463
permissions
 ACI syntax, 173
 allowing or denying access, 181
 assigning rights, 181
 overview, 180
 precedence rule, 170
plug-in functions, 471
plug-ins
 7-bit check plug-in, 471
 ACL plug-in, 471
 ACL preoperation plug-in, 472
 binary syntax plug-in, 472
 Boolean syntax plug-in, 473
 case exact string syntax plug-in, 473
 case ignore string syntax plug-in, 474
 chaining database plug-in, 474
 Class of Service plug-in, 475
 CLEAR password storage plug-in, 480
 Country String Syntax Plug-in, 475
 CRYPT password storage plug-in, 480
 disabling, 490
 distinguished name syntax plug-in, 476
 enabling, 490

- generalized time syntax plug-in, 476
- integer syntax plug-in, 477
- internationalization plug-in, 477
- ldbm database plug-in, 478
- legacy replication plug-in, 478
- multimaster replication plug-in, 479
- NS-MTA-MD5 password storage plug-in, 481
- octet string syntax plug-in, 479
- postal address string syntax plug-in, 483
- PTA plug-in, 483
- reference, 471
- referential integrity plug-in, 484
- retro changelog plug-in, 485
- roles plug-in, 486
- SHA password storage plug-in, 482
- space insensitive string syntax plug-in, 486
- SSHA password storage plug-in, 482
- state change plug-in, 487
- telephone syntax plug-in, 488
- uid uniqueness plug-in, 488
- URI plug-in, 489
- pointer CoS
 - example, 145
 - overview, 145
- port number
 - Directory Server configuration, 9
 - for SSL communications, 10
- precedence rule
 - ACI, 170
- preferences
 - security, 412
- presence index, 363
 - defaults, 367
- presence search
 - example, 562
 - syntax, 560
- preventing monopolization of the consumer in multi-master replication, 297
- pronunciation subtype, 23
- Property Editor
 - displaying, 18
- protocol data units. See PDUs, 453
- proxy authorization
 - ACI example, 234
 - with cascading chaining, 96
- proxy DN, 235

- proxy right, 182
- PTA plug-in
 - configuring, 495
 - examples, 499
 - syntax, 492
 - use in Directory Server, 491

Q

- quotation marks, in parameter values, 31

R

- read right, 181
- read-only mode, 446
 - database, 61
- read-only replica, 267
- read-write replica, 267
- redhat-directory.mib, 457
 - entity table, 460
 - entries table, 460
 - interaction table, 461
 - operations table, 458
- ref attribute, 108
- refer command, 105
- referential integrity
 - attributes, 42
 - disabling, 43
 - enabling, 43
 - log file, 42
 - modifying attributes, 44
 - overview, 41
 - with replication, 42
- referral mode, 105
- referral object class, 108
- referrals
 - creating smart referrals, 107
 - creating suffix, 109
 - on update, 55
 - setting default, 106
 - suffix, 54
- renaming entries
 - restrictions, 36
- repl-monitor.pl script, 340
- replacing attribute values, 36
- replica
 - exporting to LDIF, 326
 - read-only, 267

- read-write, 267
- replicate_now.sh script, 330
- replication
 - and access control, 241
 - and password policy, 260
 - and referential integrity, 42
 - and SSL, 332
 - cascading, 298
 - changelog, 268
 - compatibility with earlier versions, 269
 - configuring from the command line, 311
 - configuring legacy replication, 335
 - configuring SSL, 333
 - consumer server, 268
 - creating the supplier bind DN, 275
 - forcing synchronization, 329
 - hub, 268
 - managing, 267
 - monitoring status, 339
 - multi-master, 286
 - of ACIs, 241
 - overview, 267
 - replicate_now.sh script, 330
 - replicating account lockout attributes, 331
 - replication manager entry, 268
 - single-master, 276
 - solving conflicts, 342
 - supplier bind DN, 269
 - supplier server, 268
 - supplier-initiated, 268
 - troubleshooting, 347
 - unit of, 267
 - using repl-monitor.pl script, 340
 - using template-cl-dump.pl script, 347
- replication agreement, 269
- replication manager, 268
- required attributes
 - creating, 360
 - deleting, 360
 - editing, 361
- requiredObjectClass keyword, 510
- resource limits, 264
 - setting
 - using command line, 265
 - using console, 264
- Resource Summary
 - viewing, 440
- resource use
 - connections, 441
 - monitoring, 440
- restoring data, 124
 - bak2db, 127
 - bak2db.pl, 127
 - dse.ldif, 129
 - from console, 126
 - replicated entries, 129
- restoring the database, 466
- retro changelog
 - and access control, 339
 - attributes, 336
 - object class, 336
 - searching, 339
 - trimming, 338
- retro changelog plug-in
 - enabling, 337
 - overview, 270
- rights
 - list of, 181
- roledn keyword, 190
- roles, 131
 - access control, 142
 - access to directory, 190
 - activating, 263
 - attributes, 139
 - editing, 136
 - filtered
 - creating, 135
 - example, 141
 - inactivating, 132
 - managed
 - creating, 134
 - example, 140
 - nested
 - creating, 136
 - example, 142
 - object classes, 139
 - overview, 131
- root DN, see Directory Manager, 12
- root DSE, 557
- root entry creation, 546
- root suffix, 48
 - creating from command line, 51
 - creating from console, 50

S

SASL

- authentication, 200
- configuring
 - KDC server, 427
- configuring authentication at startup, 429
- configuring Kerberos, 426
- identity mapping, 422
 - configuring from the Console, 424
 - configuring from the command-line, 426
 - default, 424
- KDC server
 - configuration example, 428
- Kerberos realms, 427
- mechanisms, 421
 - CRAM-MD5, 421
 - DIGEST-MD5, 421
 - GSS-API, 421
- password change extended operation, 256

schema

- checking, 362
- creating new attributes, 355
- creating new object classes, 359
- deleting attributes, 356
- deleting object classes, 361
- differences between Directory Server and Active Directory, 534
 - initials, 535
 - street and streetAddress, 534
- editing attributes, 356
- editing object classes, 360
- extending, 353
- nsslapd-schemacheck attribute, 362
- standard, 353
- viewing attributes, 353
- viewing object classes, 357

schema checking

- and access control, 177
- ldapmodify and, 27
- overview, 362
- turning on or off, 362
- turning on or off in the command line, 362

scripts

- repl-monitor.pl, 340
- template-cl-dump.pl, 347

search filters, 559

- Boolean operators, 562
- contained in file, 558
- examples, 559
- matching rule, 564
- operators in, 560
- specifying attributes, 560
- syntax, 560
- using compound, 561
- using multiple, 561

Search Performance, 389

search right, 181

search types, list of, 560

searches

- approximate, 560
- equality, 560
- example, 556
- greater than or equal to, 560
- international, 563
- international examples, 568
- less than, 568
- less than or equal to, 560
- of directory tree, 552
- presence, 560
- specifying scope, 554
- substring, 560

searching algorithm

- overview, 367

Secure Sockets Layer, see SSL, 405

security

- certificate-based authentication, 415
- LDAP URLs, 575
- setting preferences, 412

self access, 187

- LDIF example, 188

self keyword, 187

selfwrite right, 182

- example, 232

server parameters

- database
 - read-only, 446

setting access controls, 202

setting passwords, 255

simple authentication, 199

Simple Authentication and Security Layer, 421

Simple Authentication and Security Layer (SASL). See SASL authentication, 200

- Simple Network Management Protocol. See SNMP, 453
- Simple Sockets Layer. See SSL, 200
- single-master replication
 - introduction, 270
 - setting up, 276
- smart referrals
 - creating, 107
 - creating from command line, 108
 - creating from console, 107
- SNMP
 - configuring
 - Directory Server, 457
 - ldap-agent, 455
 - managed device, 453
 - managed objects, 453
 - master agent, 453
 - configuring, 454
 - mib, 456
 - MIB
 - entity table, 460
 - entries table, 460
 - interaction table, 461
 - operations table, 458
 - monitoring the Directory Server, 453
 - overview, 453
 - subagent, 453
 - configuration file, 454
 - location, 454
 - starting, 455
 - stopping, 456
 - testing the subagent, 456
- SSL
 - Administration Server password file, 411
 - and replication, 332
 - authentication, 405
 - certificate password, 410
 - chaining with, 85
 - client authentication, 417
 - configuring clients to use, 417
 - enabling, 405
 - port number, 10
 - setting preferences, 412
 - starting the server with, 405
- SSL authentication, 200
- standard
 - attributes, 353
 - index files, 367
 - object classes, 353
 - schema, 353
- Start TLS, 394
- Starting and stopping
 - Directory Server and Administration Server, 4
 - Directory Server Console, 7
- starting the Directory Server
 - with SSL, 405
- static groups, 165
 - creating, 166
 - modifying, 166
- sub suffix, 48
 - creating from command line, 51
 - creating from console, 50
- substring index, 364
- substring index limitation, 364
- substring search, 560
 - international example, 569
- subtree level password policy, 243
- subtypes
 - of attributes, 22
- suffix
 - and associated database, 47
 - configuration attributes, 52
 - creating, 15
 - creating from command line, 51
 - creating root suffix, 50
 - creating sub suffix, 50
 - custom distribution function, 60
 - custom distribution logic, 60
 - disabling, 55
 - in Directory Server, 47
 - using referrals, 54
 - on update only, 55
 - with multiple databases, 59
- suffix referrals
 - creating, 109
 - creating from command line, 110
 - creating from console, 109
- supplier bind DN, 269
- supplier server, 268
- symbols
 - ", in ldapsearch, 553
 - , in change operation, 33
 - ::, in LDIF statements, 541

- <, in LDIF statements, 541
- quotation marks, in ldapmodify commands, 31
- synchronization agreement
 - changing, 533
- syntax
 - ACI statements, 173
 - attribute value, 356
 - LDAP URLs, 571
 - ldapsearch, 553
 - LDIF update statements, 33
 - matching rule filter, 564
 - search filter, 560
- system connections
 - monitoring, 441
- system indexes, 367
- system resources
 - monitoring, 440

T

- targattrfilters keyword, 179
- target
 - ACI syntax, 173
 - attribute values, 179
 - attributes, 176
 - keywords in ACIs, 174
 - overview, 173
 - using LDAP search filters, 178
 - using LDAP URLs, 187
- target DNs
 - containing commas, 175
- target keyword, 175
- targetattr keyword, 176
- targetfilter keyword, 178
- targeting
 - directory entries, 175
- template entry. See CoS template entry., 145
- template-cl-dump.pl script, 347
- thread
 - monitoring, 440
- time format, 578
- timeofday keyword, 198
- tuning performance
 - database, 464
 - server, 463

U

- unique attribute plug-in, 503
 - configuring, 507
 - disabling, 509
 - enabling, 509
 - examples, 511
 - markerObjectClass, 510
 - requiredObjectClass, 510
 - syntax, 504
- user access, 186
 - example, 219
 - LDIF example, 188
 - to child entries, 187
 - to own entry, 187
 - LDIF example, 188
- user and group management
 - referential integrity, 41
- user level password policy, 243
- user passwords, 255
- user-defined attributes, 354
- user-defined object classes, 357
- userattr keyword, 191
 - restriction on add, 195
- userdn keyword, 186
- users
 - activating, 263
 - inactivating, 261
- UTF-8, 577

V

- value-based ACI, 179
- viewing
 - access control
 - get effective rights, 211
 - attributes, 353
 - object classes, 357
- virtual list view index, 364
- vlvindex command-line tool, 364

W

- wildcard
 - in LDAP URL, 188
 - in target, 176
- wildcards
 - in international searches, 566

- in matching rule filters, 566
- WinSync, 515
 - about, 515
 - changing the sync agreement, 533
 - checking sync status, 533
 - configuring, 518
 - deleting entries, 531
 - groups, 530
 - logging levels, 536
 - manually updating, 532
 - Password Sync service, 521, 535
 - modifying, 535
 - setting up SSL, 523
 - starting and stopping, 535
 - uninstalling, 536
 - resurrecting deleted entries, 532
 - schema differences, 534
 - troubleshooting, 536
 - users, 528
 - using, 527
- write performance, 388
- write right, 181