Administrator's Guide

iPlanet Directory Server

Version 5.0

816-0799-01
April 2001

Contents

Introduction . . . 19
iPlanet Directory Server 5.0 Overview . . . 19
Prerequisite Reading . . . 20
Conventions Used in This Book . . . 21
Related Information . . . 21

Administering iPlanet Directory Server . . . 23

Chapter 1 Introduction to iPlanet Directory Server . . . 25
Overview of Directory Server Management . . . 26
Using the Directory Server Console . . . 26
Configuring the Directory Manager . . . 27
Binding to the Directory From iPlanet Console . . . 28
Starting and Stopping the Directory Server . . . 29
Configuring LDAP Parameters . . . 30
Starting the Server with SSL Enabled . . . 33
Cloning a Directory Server . . . 34
Starting the Server in Referral Mode . . . 35

Chapter 2 Creating Directory Entries . . . 37
Managing Entries From the Directory Console . . . 37
Managing Entries From the Command Line . . . 47
    Providing Input From the Command Line . . . 47
    Adding and Modifying Entries Using ldapmodify . . . 49
LDIF Update Statements . . . 54
        A Note on Renaming Entries . . . 58
        Adding Attributes to Existing Entries Using LDIF . . . 59
    Deleting an Entry Using LDIF . . . 63
Maintaining Referential Integrity . . . 64
    How Referential Integrity Works . . . 64

Using Referential Integrity with Replication 65
Configuring the Supplier Server 65
From the Directory Server Console 66
From the Directory Server Console 67
From the Directory Server Console 68
From the Directory Server Console 68

Chapter 3 Configuring Directory Databases 71
Creating and Maintaining Suffixes 71
Creating Suffixes 72
Maintaining Suffixes 78
Creating and Maintaining Databases 81
Creating Databases 81
Maintaining Directory Databases 86
Creating and Maintaining Database Links 87
Configuring the Chaining Policy 88
Creating a New Database Link 94
Chaining Using SSL 104
Maintaining Database Links 105
Database Links and Access Control Evaluation 106
Advanced Feature: Tuning Database Link Performance 108
Detecting Errors During Normal Processing 111
Managing Threaded Operations 112
Advanced Feature: Configuring Cascading Chaining 113
Overview of Cascading Chaining 113
Summary of Cascading Chaining Configuration Attributes 120
Cascading Chaining Configuration Example 121
Using Referrals 125
Setting Default Referrals 126
Creating Smart Referrals 127
Creating Suffix Referrals 129

Chapter 4 Populating Directory Databases 133
Importing Data 133
Importing From the Command Line 137
Exporting Data 140
Backing Up and Restoring Data 144
Backing Up All Databases 145
Backing Up the dse.ldif Configuration File 147
Restoring All Databases 147
Restoring Databases that Include Replicated Entries 150
Restoring the dse.ldif Configuration File 151

Enabling and Disabling Read-Only Mode 151

Chapter 5 Advanced Entry Management 153

Using Groups 153

  Managing Static Groups 154

  Managing Dynamic Groups 155

Using Roles 156

  About Roles 156

  Managing Roles Using the Console 157

  Managing Roles Using the Command Line 162

    Examples: Managed Role Definition 163

    Example: Filtered Role Definition 164

    Example: Nested Role Definition 165

  Using Roles Securely 165

Assigning Class of Service 166

  About CoS 167

    About the CoS Definition Entry 167

    About the CoS Template Entry 168

    How a Pointer CoS Works 169

    How an Indirect CoS Works 169

    How a Classic CoS Works 170

  Managing CoS Using the Console 171

  Managing CoS From the Command Line 174

    Example of a Pointer CoS 178

    Example of an Indirect CoS 179

    Example of a Classic CoS 180

  Creating Role-Based Attributes 181

  Access Control and CoS 182

Chapter 6 Managing Access Control 183

Access Control Principles 184

  ACI Structure 184

  ACI Placement 185

  ACI Evaluation 185

  ACI Limitations 186

Default ACIs 187

Creating ACIs Manually 188

  The ACI Syntax 188

    Example ACI 189

    Targeting Attributes 192

    Rights Required for LDAP Operations 198

    Permissions Syntax 199

Bind Rules 200
Bind Rule Syntax 200
Anonymous Access (anyone Keyword) 202
General Access (all Keyword) 203
Self Access (self Keyword) 203
Parent Access (parent Keyword) 203
LDAP URLs 203
Wildcards 204
Examples 204
Examples 206
Examples 215
Examples 216
Creating ACIs From the Console 218
Access Control Usage Examples 224
Granting Anonymous Access 225
Granting Write Access to Personal Entries 227
Restricting Access to Key Roles 231
Granting a Group Full Access to a Suffix 232
Granting Rights to Add and Delete Group Entries 233
Granting Conditional Access to a Group or Role 235
Denying Access 238
Setting a Target Using Filtering 240
Allowing Users to Add or Remove Themselves From a Group 240
Defining Permissions for DNs That Contain a Comma 242
Proxied Authorization ACI Example 242
Viewing the ACIs for an Entry 243
Advanced Access Control: Using Macro ACIs 244
Macro ACI Example 244
Macro ACI Syntax 247
Macro Matching for ($dn) 248
Macro Matching for [$dn] 248
Macro Matching for ($attr.*attrName*) 249
Access Control and Replication 250
Logging Access Control Information 251
Compatibility with Earlier Releases 251

Chapter 7 User Account Management 253
Managing the Password Policy 253
Configuring the Password Policy 254
Configuring the Password Policy Using the Console 254
Configuring the Password Policy Using the Command-Line 255
Setting User Passwords 259
Configuring the Account Lockout Policy 260

Configuring the Account Lockout Policy Using the Console 260
Configuring the Account Lockout Policy Using the Command Line 260
Managing the Password Policy in a Replicated Environment 262
Inactivating Users and Roles 263
Inactivating User and Roles Using the Console 264
Inactivating User and Roles Using the Command Line 264
Activating User and Roles Using the Console 265
Activating User and Roles Using the Command Line 266
Setting Resource Limits Based on the Bind DN 266
Setting Resource Limits Using the Console 267
Setting Resource Limits Using the Command Line 267

Chapter 8 Managing Replication 269
Replication Overview 270
Read-Write Replica / Read-Only Replica 270
Supplier / Consumer 270
Change Log 271
Unit of Replication 271
Replication Identity 271
Replication Agreement 272
Compatibility with Earlier Versions of Directory Server 272
Replication Scenarios 273
Single-Master Replication 273
Multi-Master Replication 274
Cascading Replication 276
Configuring Single-Master Replication 277
Configuring Multiple-Master Replication 280
Configuring Cascading Replication 285
Configuration Tips 290
Detailed Procedures 291
Removing the Change Log 296
Initializing Consumers 297
When to Initialize a Consumer 297
Exporting a Replica to LDIF 300
Importing the LDIF File to the Consumer Server 300
Forcing Replication Updates 300
Replication over SSL 304
Replication with Earlier Releases 306
Using the Retro Change Log Plug-In 308
Monitoring Replication Status 311
Solving Common Replication Conflicts 312
Naming Conflicts 313
Deleted Entries with Child Entries 315

Controlling Access to Conflicting Entries 315

Chapter 9 Extending the Directory Schema 317
Overview of Extending Schema 317
Managing Attributes 318
Managing Object Classes 321
Turning Schema Checking On and Off 326

Chapter 10 Managing Indexes 327
About Indexes 327
About Index Types 328
About Default, System, and Standard Indexes 329
Overview of Default Indexes 329
Overview of System Indexes 331
Overview of Standard Indexes 331
Overview of the Searching Algorithm 332
Balancing the Benefits of Indexing 334
Creating Indexes 336
Creating Indexes From the Server Console 337
Creating Indexes From the Command Line 338
Adding an Index Entry 338
Running the db2index.pl Script 340
The following table describes the `db2index.pl` options used in the examples: 341
Creating Browsing Indexes From the Server Console 341
Creating Browsing Indexes from the Command Line 342
Adding a Browsing Index Entry 342
Running the vlvindex Script 344
Deleting Indexes 345
Deleting Indexes From the Server Console 346
Deleting Indexes From the Command Line 347
Deleting an Index Entry 347
Running the db2index.pl Script 348
Deleting Browsing Indexes From the Server Console 349
Deleting Browsing Indexes From the Command Line 350
Deleting a Browsing Index Entry 350
Running the vlvindex Script 352
Managing Indexes 353
Benefits of the All IDs Mechanism 353
Drawbacks of the All IDs Mechanism 354
When All IDs Threshold is Too Low 354
When All IDs Threshold is Too High 355
All IDs Threshold Tuning Advice for Single- Enterprise Directories 355

All IDs Threshold Tuning Advice for Service Providers and Extranets 356
Default All IDs Threshold Value 357
Symptoms of an Inappropriate All IDs Threshold Value 357
Changing the All IDs Threshold Value 358
Attribute Name Quick Reference Table 359

Chapter 11 Managing SSL 361
Introduction to SSL in the Directory Server 361
Obtaining and Installing Server Certificates 363
Activating SSL 368
Setting Security Preferences 369
Using Certificate-Based Authentication 371
Configuring LDAP Clients to Use SSL 373

Chapter 12 Monitoring Server and Database Activity 377
Viewing and Configuring Log Files 377
Access Log 379
Error Log 380
Audit Log 382
Manual Log File Rotation 384
Monitoring Server Activity 384
Viewing the Server Performance Monitor 385
Overview of Server Performance Monitor Information 385
General Information (Server) 385
Resource Summary 386
Current Resource Usage 387
Connection Status 387
Global Database Cache Information 388
Monitoring Database Activity 391
Viewing Database Performance Monitors 391
Overview of Database Performance Monitor Information 392
General Information (Database) 392
Summary Information Table 392
Database Cache Information Table 393
Database File-Specific Table 394
Monitoring Database Link Activity 397

Chapter 13 Monitoring Directory Server Using SNMP 399
About SNMP 399
SNMP Overview 400
NMS-Initiated Communication 401
Managed Device-Initiated Communication 401

Overview of the Directory Server Management Information Base 402
About the Operations Table 402
The Entries Table 404
Setting Up SNMP 405
Setting Up SNMP on Windows NT 405
Setting Up SNMP on UNIX 405
Configuring the AIX SNMP Daemon 406
Starting and Stopping the SNMP Subagent on UNIX 407
Starting and Stopping the SNMP Service on Windows NT 407
Configuring SNMP for the Directory Server 408

Chapter 14 Tuning Directory Server Performance 409
Tuning Server Performance 409
Tuning Database Performance 410
Optimizing Search Performance 410
Tuning Transaction Logging 413
Changing the Location of the Database Transaction Log 414
Changing the Database Checkpoint Interval 414
Disabling Durable Transactions 415

iPlanet Plug-Ins Reference 417

Chapter 15 Administering Directory Server Plug-Ins 419
Server Plug-in Functionality Reference 419
7-bit Check Plug-In 420
ACL Plug-In 421
ACL Preoperation Plug-In 421
Binary Syntax Plug-In 422
Boolean Syntax Plug-In 422
Case Exact String Syntax Plug-In 423
Case Ignore String Syntax Plug-In 423
Chaining Database Plug-In 424
Class of Service Plug-In 424
Country String Syntax Plug-In 425
Distinguished Name Syntax Plug-In 425
Generalized Time Syntax Plug-In 426
Integer Syntax Plug-In 427
Internationalization Plug-In 427
ldbm Database Plug-In 428
Legacy Replication Plug-In 428
Multimaster Replication Plug-In 429
Octet String Syntax Plug-in 429
CLEAR Password Storage Plug-In 430

CRYPT Password Storage Plug-In 430
NS-MTA-MD5 Password Storage Plug-In 431
SHA Password Storage Plug-In 432
SSHA Password Storage Plug-in 432
Postal Address String Syntax Plug-In 433
PTA Plug-In 433
Referential Integrity Postoperation Plug-In 434
Retro Change Log Plug-In 435
Roles Plug-In 435
Telephone Syntax Plug-In 436
UID Uniqueness Plug-in 437
URI Plug-in 439
Enabling and Disabling Plug-Ins From the Server Console 439

Chapter 16 Using the Pass-Through Authentication Plug-In 441
How Directory Server 5.0 Uses PTA 441
PTA Plug-In Syntax 443
Configuring the PTA Plug-In 445
PTA Plug-In Syntax Examples 451

Chapter 17 Using the Attribute Uniqueness Plug-In 455
Overview of the Attribute Uniqueness Plug-In 455
Overview of the UID Uniqueness Plug-in 457
Attribute Uniqueness Plug-In Syntax 457
Creating an Instance of the Attribute Uniqueness Plug-In 460
Configuring Attribute Uniqueness Plug-Ins 461
Configuring Attribute Uniqueness Plug-Ins From the Directory Server Console 461
Attribute Uniqueness Plug-In Syntax Examples 465
Replication and the Attribute Uniqueness Plug-In 467
Simple Replication Scenario 467
Multi-Master Replication Scenario 467

Appendixes 469

Appendix A LDAP Data Interchange Format 471
LDIF File Format 471
Continuing Lines in LDIF 473
Representing Binary Data 473
Specifying Directory Entries Using LDIF 475
Defining Directories Using LDIF 480
LDIF File Example 481
Storing Information in Multiple Languages 482

Appendix B Finding Directory Entries 485
Finding Entries Using the Server Console 485
Using ldapsearch 486
Using Special Characters 486
ldapsearch Command-Line Format 487
Commonly Used ldapsearch options 487
ldapsearch Examples 489
Returning All Entries 490
Specifying Search Filters on the Command Line 490
Searching the Root DSE Entry 490
Searching the Schema Entry 490
Using LDAP_BASEDN 491
Displaying Subsets of Attributes 491
Specifying Search Filters Using a File 491
Specifying DNs that Contain Commas in Search Filters 492
Using Client Authentication When Searching 492
LDAP Search Filters 493
Search Filter Syntax 493
Using Attributes in Search Filters 494
Using Operators in Search Filters 494
Using Compound Search Filters 496
Search Filter Examples 496
Searching an Internationalized Directory 497
Matching Rule Filter Syntax 498
Matching Rule Formats 498
Using Wildcards in Matching Rule Filters 500
Supported Search Types 501
International Search Examples 502
Less Than Example 502
Less Than or Equal to Example 502
Equality Example 503
Greater Than or Equal to Example 503
Greater Than Example 503
Substring Example 503

Appendix C LDAP URLs 505
Components of an LDAP URL 505
Escaping Unsafe Characters 507
Examples of LDAP URLs 508

Appendix D Internationalization 511
About Locales 511
Identifying Supported Locales 513

Supported Language Subtypes 515

Glossary 517

Index 533

List of Figures

Figure 1-1 Viewing the Bind DN 29
Figure 2-1 Directory Server Console - Property Editor 42
Figure 6-1 Using Inheritance With the `userattr` Keyword 211
Figure 6-2 Selecting an Object in the Navigation Tree to Set Access Control 220
Figure 6-3 Access Control Editor Window 221
Figure 6-4 Example directory tree for Macro ACIs 245
Figure 8-1 Single-Master Replication 274
Figure 8-2 Multi-Master Replication 275
Figure 8-3 Cascading Replication 276

List of Tables

Table 2-1 Entry Templates and Corresponding Object Classes 39
Table 3-1 Suffix Attributes 76
Table 3-2 Components Allowed to Chain 89
Table 3-3 LDAP Controls and Their OIDs 93
Table 3-4 Database Link Configuration Attributes 100
Table 3-5 Database Link Connection Management Attributes 110
Table 3-6 Database Link Processing Error Detection Parameters 111
Table 3-7 Cascading Chaining Configuration Attributes 120
Table 4-1 Import Method Comparison 134
Table 5-1 CoS Definition Entry Object Classes 174
Table 5-2 CoS Definition Entry Attributes 175
Table 5-3 CoS Definitions 176
Table 6-1 LDIF Target Keywords 190
Table 6-2 LDIF Bind Rule Keywords 201
Table 6-3 Macros in ACI Keywords 247
Table 7-1 Password Policy Attributes 256
Table 7-2 Account Lockout Policy Attributes 261
Table 8-1 Replicate_Now Variables 304
Table 8-2 Attributes of a Retro Change Log Entry 309
Table 8-3 Directory Server Console - Status Tab 312
Table 9-1 Attributes Tab Reference 318
Table 9-2 Object Classes Tab Reference 322
Table 10-1 Default indexes 330
Table 10-2 System indexes 331
Table 10-3 Attribute Name Quick Reference Table 359
Table 12-1 Server Performance Monitoring - Resource Summary Table 386
Table 12-2 Server Performance Monitoring - Current Resource Usage Table 387

Table 12-3 Server Performance Monitoring - Connection Status Table 388
Table 12-4 Server Performance Monitoring - Global Database Cache Table 388
Table 12-5 Database Performance Monitoring - Summary Information 392
Table 12-6 Database Performance Monitoring - Database Cache Information 393
Table 12-7 Database Performance Monitoring - Database File-Specific table 394
Table 12-8 Database Link Monitoring Attributes 397
Table 13-1 Operations Table Managed Objects and Descriptions 403
Table 13-2 Entries Table Managed Objects and Descriptions 404
Table 16-1 PTA Plug-In Parameters 444
Table 17-1 Attribute Uniqueness Plug-In Variables 459
Table A-1 LDIF Fields 472
Table A-2 LDIF Elements in Organization Entries 476
Table A-3 LDIF Elements in Organizational Unit Entries 477
Table A-4 LDIF Elements in Person Entries 479
Table B-1 Search Filter Operators 494
Table B-2 Search Filter Boolean Operators 496
Table B-3 Search Types, Operators, and Suffixes 501
Table C-1 LDAP URL Components 505
Table D-1 Supported Locales 513
Table D-2 Supported Language Subtypes 515

Introduction

iPlanet Directory Server 5.0 is a powerful and scalable distributed directory server based on the industry-standard Lightweight Directory Access Protocol (LDAP). iPlanet 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 iPlanet Directory Server.

iPlanet Directory Server 5.0 Overview

iPlanet Directory Server 5.0 provides the following key features:

- Multi-master replication—Provides a highly available directory service for both read and write operations. Multi-master replication can be combined with simple and cascading replication scenarios to provide a highly flexible and scalable replication environment.
- Chaining and referrals—Increases the power of your directory by storing a complete logical view of your directory on a single server while maintaining data on a large number of directory servers, transparently for clients.
- Roles and Class of Service—Provides a flexible mechanism for grouping and sharing attributes between entries in a dynamic fashion.
- Improved access control mechanism—Provides support for macros that dramatically reduce the number of access control statements used in the directory, and increase the speed of access control evaluation.
- Resource-limits by bind DN—Gives you the power to control the amount of server resources allocated to search operations based on the bind DN of the client.
- Multiple databases—Provides a simple way of breaking down your directory data to simplify the implementation of replication and chaining in your directory service.

- Password Policy and Account Lockout—Allows you to define a set of rules that govern how passwords and user accounts are managed in the directory server.
- SSL—Provides secure communications over the network including ciphers with up to 168-bit encryption.

The major components of iPlanet Directory Server 5.0 include:

- An LDAP server—The core of the directory service, provided by the `slapd` daemon, and compliant with the LDAP v3 Internet standards.
- Directory Server Console—An improved management console that dramatically reduces the effort of setting up and maintaining your directory service. The directory console is part of iPlanet Console, the common management framework for iPlanet servers.
- Directory Server Gateway—A customizable HTTP to LDAP client that allows you to access directory data from a web browser.
- iPlanet Directory Express—A simple directory lookup tool that you can use right out of the box.
- SNMP Agent—Permits you to monitor your directory server in real time using the Simple Network Management Protocol (SNMP).
- Online backup and restore—Allows you to create backups and restore from backups while the server is running.

Prerequisite Reading

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

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

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

Conventions Used in This Book

This section explains the conventions used in this book.

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

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

The greater than symbol (>) is used as a separator for successive menu selections. For example, Object > New > User means that you should pull down the Object menu, drag the mouse down to highlight New, and drag the mouse across to the New submenu in which you must select User.

Throughout this book you will see path references of the form:

`/usr/iplanet/servers/slapd-`*`serverID`*`/...`

The `/usr/iplanet/servers` directory is the default installation directory. If you have installed the Directory Server in a different location, you should adapt the path accordingly. *serverID* represents the server identifier you gave the server when you installed it. For example, if you gave the server an identifier of **phonebook**, then the actual path would be:

`/usr/iplanet/servers/slapd-phonebook/. . .`

All paths specified in this manual are in UNIX format. If you are using a Windows NT-based directory server, use equivalent paths.

Related Information

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

iPlanet Directory Server Installation Guide. Procedures for installing your Directory Server as well as procedures for migrating your Netscape Directory Server to iPlanet Directory Server.

iPlanet Directory Server Deployment Guide. Provides an overview for planning your deployment of the Directory Server. Includes deployment examples.

iPlanet Directory Server Configuration, Command, and File Reference. Provides reference information on the command-line scripts, configuration attributes, and log files shipped with Directory Server.

iPlanet Schema Reference. Information about all the schema used in the iPlanet suite of products.

Other useful iPlanet information can be found at the following Internet locations:

- iPlanet release notes and other documentation:
  http://docs.iplanet.com/docs/manuals/
- iPlanet product status:
  http://www.iplanet.com/support/technical_resources/
- iPlanet Professional Services information:
  http://www.iplanet.com/services/pro_serv/index.html
- iPlanet developer information:
  http://developer.iplanet.com/
- iPlanet learning solutions:
  http://www.iplanet.com/learning/index.html
- iPlanet product data sheets:
  http://www.iplanet.com/products/index.html

Administering iPlanet Directory Server

Chapter 1, “Introduction to iPlanet Directory Server”

Chapter 2, “Creating Directory Entries”

Chapter 3, “Configuring Directory Databases”

Chapter 4, “Populating Directory Databases”

Chapter 5, “Advanced Entry Management”

Chapter 6, “Managing Access Control”

Chapter 7, “User Account Management”

Chapter 8, “Managing Replication”

Chapter 9, “Extending the Directory Schema”

Chapter 10, “Managing Indexes”

Chapter 11, “Managing SSL”

Chapter 12, “Monitoring Server and Database Activity”

Chapter 13, “Monitoring Directory Server Using SNMP”

Chapter 14, “Tuning Directory Server Performance”

Chapter 1

Introduction to iPlanet Directory Server

iPlanet Directory Server product ships with a Directory Server, an administration server, and iPlanet Console. This chapter provides overview information about the Directory Server, and the most basic tasks you need to start administering a directory service.

It includes the following sections:

- Overview of Directory Server Management
- Using the Directory Server Console
- Binding to the Directory From iPlanet Console
- Starting and Stopping the Directory Server
- Configuring the Directory Manager
- Configuring LDAP Parameters
- Starting the Server with SSL Enabled
- Cloning a Directory Server
- Starting the Server in Referral Mode

Overview of Directory Server Management

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

You perform most Directory Server administrative tasks through the Administration Server, a second server that iPlanet provides to help you manage Directory Server (and all other iPlanet servers). For Directory Server, you use a part of the Administration Server called iPlanet Console. *Directory Server Console* is a part of iPlanet Console designed specifically for use with iPlanet Directory Server.

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

Using the Directory Server Console

The Directory Server Console is an integral part of the iPlanet Console. You start the Directory Server Console from iPlanet Console, which is described in *Managing Servers with iPlanet Console*.

Starting Directory Server Console

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

```
# /usr/iplanet/servers/slapd-serverID/start-slapd
```

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

```
# /usr/iplanet/servers/start-admin
```

3. Start iPlanet Console by entering the following command:

```
% /usr/iplanet/servers/startconsole
```

The Console login window is displayed. Or, if your configuration directory (the directory that contains the `o=NetscapeRoot` suffix) is stored in a separate instance of Directory Server, a window is displayed requesting the administrator user id, password, and the URL of the Admin Server for that directory server.

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

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

5. On the Topology tab, go down the navigation tree until you locate the Directory Server icon, and double-click this icon.

   The Directory Server Console is displayed.

Configuring the Directory Manager

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

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

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

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

   If you are already logged in to the Console, see “Binding to the Directory From iPlanet Console,” on page 28 for instructions on how to log in as a different user.

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

   The default value is `cn=Directory Manager`.

4. 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.
5. Enter the new password and confirm it using the text fields provided.
6. Click Save.

Binding to the Directory From iPlanet Console

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

Changing Login Identity

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

To change your login in iPlanet Console:

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

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

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

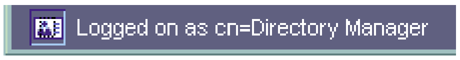
   ```
   cn=Directory Manager
   ```

For more information about the Directory Manager DN and password, refer to “Configuring the Directory Manager,” on page 27.

Viewing the Current Bind DN From the Console

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

Figure 1-1 Viewing the Bind DN


Logged on as cn=Directory Manager

Starting and Stopping the Directory Server

If you are not using Secure Sockets Layer (SSL), you can start and stop the Directory Server using the methods listed here. If you are using SSL, see "Starting the Server with SSL Enabled," on page 33.

NOTE	On UNIX systems, rebooting the system does not automatically start the `slapd` process. This is because the directory does not automatically create startup or run command (`rc`) scripts. Check your operating system documentation for details on adding these scripts.

Starting/Stopping the Server From the Console

1. Start the Directory Server Console.

   For instructions, refer to "Starting Directory Server Console," on page 26.

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

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

Alternatively, if you are using a Windows NT machine, from the Windows NT Services Control Panel:

1. Select Start > Settings > Control Panel from the desktop.

2. Double-click the Services icon.
3. Scroll through the list of services and select the iPlanet Directory Server.

   The service name is `iPlanet Directory Server 5.0 (`*`serverID`*`)`, where *serverID* is the identifier you specified for the server when you installed it.
4. Start or stop the service:
   - To stop the service, click Stop and then confirm that you want to stop the service.
   - To start the service, select the Directory Server service and click Start.

Starting/Stopping the Server From the Command Line

Use one of the following scripts:

```
/usr/iplanet/servers/slapd-serverID/start-slapd
```

or

```
/usr/iplanet/servers/slapd-serverID/stop-slapd
```

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

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

Configuring LDAP Parameters

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

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

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

Changing Directory Server Port Numbers

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

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

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

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

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

1. On the Directory Server Console, select the Configuration tab and then select the top entry in the navigation tree in the left pane.
2. Select the Settings tab in the right pane.
3. Enter the port number you want the server to use for non-SSL communications in the “Port” text box.

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

   The encrypted port number that you specify must not be the same port number as you are using for normal LDAP communications. The default value is 636.
5. Click Save and then restart the server.

   See “Starting and Stopping the Directory Server,” on page 29 for information.

Placing the Entire Directory Server in Read-Only Mode

If you maintain more than one database with your directory server and you need to place all your databases in read-only mode, you can do this in a single operation. Note, however, that if your Directory Server contains replicas, you must not use read-only mode because it will disable replication.

To put the Directory Server in read-only mode:

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

NOTE	This operation also makes the directory server configuration read-only; therefore, you cannot update the server configuration, enable or disable plug-ins, or even restart the directory server while it is in read-only mode.

For information on placing a single database in read-only mode, refer to “Enabling Read-Only Mode,” on page 152.

Tracking Modifications to Directory Entries

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

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

NOTE When a database link is used by a client application to create or modify entries, the `creatorsName` and `modifiersName` attributes do not reflect the real creator or modifier of the entries. These attributes contain the name of the administrator who is granted proxy authorization rights on the remote server. For information on proxy authorization, refer to "Providing Bind Credentials," on page 97.

To enable the Directory Server to track this information:

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

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

   See "Starting and Stopping the Directory Server," on page 29 for more information.

Starting the Server with SSL Enabled

On Windows NT, if you are using SSL with your server, you must start the server from the server's host machine. This is because a dialog box will prompt you for the certificate PIN before the server will start. For security reasons, this dialog box appears only on the server's host machine.

On UNIX, you must start the server from the command line.

Alternatively, on either platform, you can create a password file to store your certificate password. By placing your certificate database password in a file, you can start your server from the server console, and also allow your server to automatically restart 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 your server is running in an unsecured environment.

The password file must be placed in the following location:

`/usr/iplanet/servers/alias/slapd-`*`serverID`*`-pin.txt`

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

You need to include the token name and password in the file as follows:

```
Token:Password
```

For example:

```
Internal (Software) Token:mypassword
```

To create certificate databases, you must use the administration server and the Certificate Setup Wizard. For information on certificate databases, certificate aliases, SSL, and obtaining a server certificate, see *Managing Servers with iPlanet Console*. For information on using SSL with your Directory Server, see Chapter 11, “Managing SSL.”

Cloning a Directory Server

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

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

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

Creating a New Directory Server Instance

1. In the iPlanet Console window, select then right click Server Group in the navigation tree.
2. From the pop-up menu, select Create Instance of > Directory Server.

   The Create New Instance dialog box is displayed.

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

   This name must not include the period (.) symbol.

4. Enter the a port number for LDAP communications in the Network port field.

5. Enter the suffix managed by this new instance of the directory in the base suffix field.

6. Enter a DN for the Directory Manager in the Root DN field.

   For information on the role and privileges of the Directory Manager entry, refer to “Configuring the Directory Manager,” on page 27.

7. Enter the password for this user in the Password for Root DN field, and confirm it by entering it again in the Confirm Password field.

8. If running the server on a UNIX host, enter the user ID for the directory server daemon, in the Server Runtime User ID field.

9. Click OK.

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

Cloning the Directory Configuration

1. In the iPlanet Console window, expand the Server Group folder, and right-click on the directory server that you want to clone.

2. From the pop-up menu, select Clone Server Config.

   A new window is displayed with the list of target servers for cloning.

3. In this window, select the server to which you want the configuration to apply, and click the Clone To button.

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

Starting the Server in Referral Mode

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

If the server is already running, you can put it in referral mode by using the Directory Server Console. This procedure is explained in “Setting Default Referrals,” on page 126.

Using the refer Command

On a UNIX machine, to start the Directory Server in referral mode follow these steps:

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

   ```
   prompt% cd /usr/iplanet/servers/slapd-serverID/bin/slapd/server
   ```

2. Run the `refer` command as follows:

   ```
   prompt% ./ns-slapd refer -p port -r ldapurl
   ```

   where *port* is the port number of the Directory Server you want to start in referral mode, and *ldapurl* is the referral returned to clients. For information on the format of an LDAP URL, refer to Appendix C, “LDAP URLs.”

On a Windows NT machine, to start the Directory Server in referral mode follow these steps:

1. Go to the following directory under your installation directory:

   ```
   \iplanet\servers\slapd-serverID\bin\slapd\server
   ```

2. Run the `refer` command as follows:

   ```
   slapd refer -p port -r ldapurl
   ```

   where *port* is the port number of the Directory Server you want to start in referral mode, and *ldapurl* is the referral returned to clients. For information on the format of an LDAP URL, refer to Appendix C, “LDAP URLs.”

Chapter 2

Creating Directory Entries

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

During the planning phase of your directory deployment, you should characterize the types of data that your directory will contain. You should read *iPlanet Directory Server Deployment Guide* before creating entries and modifying the default schema.

This chapter consists of the following sections:

- Managing Entries From the Directory Console
- Managing Entries From the Command Line
- LDIF Update Statements
- Maintaining Referential Integrity

Managing Entries From the Directory Console

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

If you want to add several entries simultaneously, you can use the command-line utilities described in "Managing Entries From the Command Line," on page 47.

This section provides information about:

- Creating a Root Entry
- Creating Directory Entries
- Modifying Directory Entries

- Deleting Directory Entries

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

NOTE You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for your directory, see Chapter 6, "Managing Access Control."

Creating a Root Entry

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

To create a root entry for a database:

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 26.

2. Create a new database, as explained in "Creating and Maintaining Databases," on page 81.

3. On the Directory tab, right-click the top object representing the directory server, and choose New Root Object.

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

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

   The New Object window is displayed.

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

   The object class you select must contain the attribute you used to name the suffix. For example, if you are creating the entry corresponding to the suffix `ou=people,dc=siroe,dc=com`, then you can choose the `organizationalUnit` object class (or another object class that allows the `ou` attribute).

6. Click OK in the New Object window.

The Property Editor for the new entry is displayed. You can accept the current values by clicking OK, or modify the entry, as explained in “Modifying Directory Entries,” on page 41.

Creating Directory Entries

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

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

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

Table 2-1 Entry Templates and Corresponding Object Classes

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

These templates contain fields representing all the mandatory attributes, and some of the commonly used optional attributes. To create an entry using one of these templates, refer to “Creating an Entry Using a Predefined Template,” on page 40. To create any other type of entry, refer to “Creating Other Types of Entries,” on page 40.

Creating an Entry Using a Predefined Template

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 26.

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

   The corresponding Create window is displayed.

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

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

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

   The Property Editor is displayed. Refer to "Modifying Directory Entries," on page 41 for information on using the Property Editor.

5. Click OK to dismiss the Create window.

   The new entry is displayed in the right pane.

Creating Other Types of Entries

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 26.

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

   The New Object window is displayed.

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

4. Click OK.

   If you selected an object class related to a type of entry for which a predefined template is available, the corresponding Create window is displayed. (See "Creating an Entry Using a Predefined Template," on page 40).

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

5. Supply a value for all the listed attributes.

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

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

To provide values for optional attributes that are not listed, refer to “Modifying Directory Entries,” on page 41.

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

The new entry is displayed in the right pane.

Modifying Directory Entries

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

From the Property Editor, you can:

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

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

Displaying the Property Editor

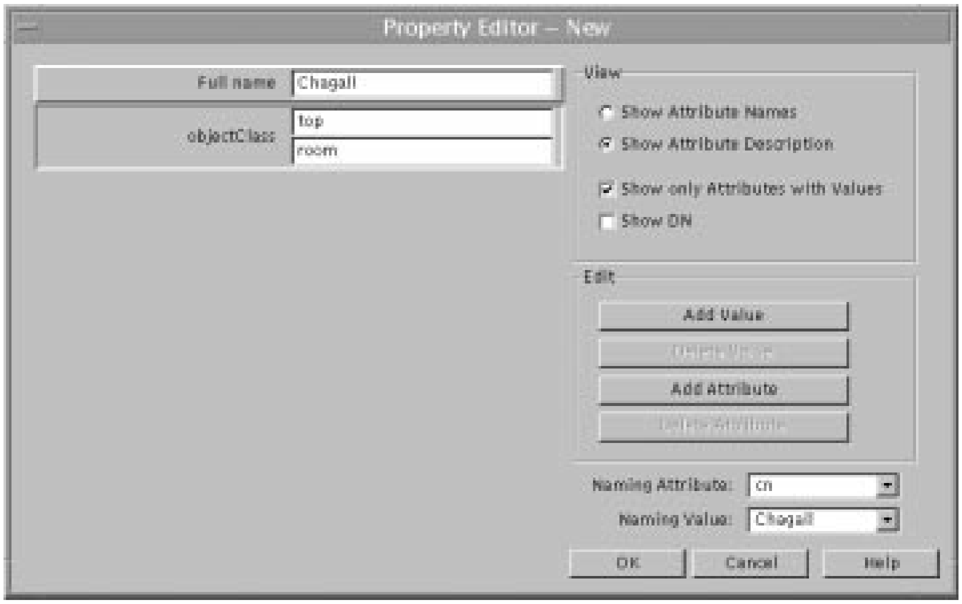
You can start the Property Editor in several ways:

- From the Directory tab, by right-clicking an entry in the left or right pane, and selecting Properties from the pop-up menu.
- From the Directory tab, by double-clicking an entry in the left or right pane.

- From the Create User, Group, Organizational Unit, Role, and Class of Service templates, by clicking the Advanced button (see "Creating an Entry Using a Predefined Template," on page 40.)
- From the New Object window, by clicking OK (see "Creating Other Types of Entries," on page 40).

The Property Editor window is shown in Figure 2-1 with an example entry that describes a room.

Figure 2-1 Directory Server Console - Property Editor



Full name: Chagall
objectClass: top, room

Show Attribute Names
[✓ Show Attribute Description]
[✓ Show only Attributes with Values]
Show DN

Naming Attribute: cn
Naming Value: Chagall

Adding an Object Class to an Entry

To add an object class to an entry:

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

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

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

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

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

4. Click OK in the Property Editor when you have finished editing the entry.

   The Property Editor is dismissed.

Removing an Object Class

To remove an object class from an entry:

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

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

3. Click OK in the Property Editor when you have finished editing the entry.

   The Property Editor is dismissed.

Adding an Attribute to an Entry

Before you can add an attribute to an entry, the entry must contain an object class that either requires or allows the attribute. See “Adding an Object Class to an Entry,” on page 42 and Chapter 9, “Extending the Directory Schema” for more information.

To add an attribute to an entry:

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

2. Click Add Attribute.

   The Add Attribute dialog box is displayed.

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

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

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

5. Click OK in the Property Editor when you have finished editing the entry.

   The Property Editor is dismissed.

Adding Attribute Values

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

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

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

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

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

3. Type in the name of the new attribute value.
4. Click OK in the Property Editor when you have finished editing the entry.

   The Property Editor is dismissed.

Removing an Attribute Value

To remove an attribute value from an entry:

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

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

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

3. Click OK in the Property Editor when you have finished editing the entry.

   The Property Editor is dismissed.

Adding an Attribute Subtype

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

Language Subtype

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

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

attribute`;lang-`*subtype*

Where *attribute* is the attribute you are adding to the entry and *subtype* is the two character abbreviation for the language. See Table D-2 on page 515 for a list of supported language subtypes. For example:

```
givenname;lang-ja
```

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

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

Instead, use:

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

Binary Subtype

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

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

Pronunciation Subtype

Assigning the pronunciation subtype to an attribute indicates that the attribute value is a phonetic representation. The subtype is added to the attribute name as follows: *attribute*`;phonetic`.

This subtype is commonly used in combination with a language subtype for languages that have more than one alphabet, where one is a phonetic representation.

You might want to use this with attributes that are expected to contain user names, such as `cn` or `givenname`. For example, `givenname;lang-ja;phonetic` indicates that the attribute value is the phonetic version of the entry's Japanese name.

To Add a Subtype Using the Property Editor:

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

   Alternatively, you can double-click the entry. The Property Editor is displayed.

2. Click Add Attribute.

   The Add Attribute dialog box displays.

3. Select the attribute you want to add from the list.
4. To assign a language subtype to the attribute, select it from the Language drop-down list.
5. From the Subtype drop-down list you can also assign one of two other subtypes: binary, or pronunciation.
6. Click OK.

   The Add Attribute window is dismissed.

7. When you have finished defining the information for the entry, click OK in the Property Editor.

Deleting Directory Entries

To delete entries using the Directory Server Console:

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

   For information on starting the Directory Server Console, refer to “Using the Directory Server Console,” on page 26.

2. Right-click the entry you want to delete in the navigation tree, or in the right pane, and select Delete from the pop-up menu.

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

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

Managing Entries From the Command Line

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

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

This section provides information about:

- Providing Input From the Command Line
- Creating a Root Entry From the Command Line
- Adding Entries Using LDIF
- Adding and Modifying Entries Using ldapmodify
- Deleting Entries Using ldapdelete
- Using Special Characters

NOTE You cannot modify your directory unless the appropriate access control rules have been set. For information on creating access control rules for your directory, see Chapter 6, “Managing Access Control.”

Providing Input From the Command Line

When you provide input to the `ldapmodify` and `ldapdelete` utilities directly from the command line, you must use LDIF statements. For detailed information on LDIF statements, refer to “LDIF Update Statements,” on page 54.

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

Typically, the EOF escape sequence is one of the following, depending upon the type of machine you use:

- UNIX—Almost always control-D (^D)

- Windows NT—Usually control-Z followed by a carriage return (^Z<return>)

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

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

When you add an entry from the command-line or from LDIF, make sure that an entry representing a subtree is created before new entries are created under that branch. For example, if you want to place an entry in a People subtree, then create entry representing that subtree before creating entries within the subtree.

For example:

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

Creating a Root Entry From the Command Line

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

```
prompt% ldapmodify -a -D "dn=directory manager" -w secret
```

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

You create the new root object as follows:

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

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

NOTE You can use this method only if you have one database per suffix. If you create a suffix that is stored in several databases, you must use the `ldif2db` utility with the `-n` option to specify the database that will hold the new entries. For information, refer to “Importing From the Command Line,” on page 137.

Adding Entries Using LDIF

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

1. Define the entries in an LDIF file.

   LDIF is described in Appendix A, “LDAP Data Interchange Format.”

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

   See “Performing an Import From the Console,” on page 134 for information. When you import the LDIF file, select “Append to database” on the Import dialog box so that the server will only import entries that do not currently exist in the directory.

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

Adding and Modifying Entries Using ldapmodify

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

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

- If the server detects an attribute or object class in the entry that is not known to the server, then the modify operation will fail when it reaches the erroneous entry. All entries that were processed before the error was encountered will be successfully added or modified. If you run `ldapmodify` with the `-c` option (do not stop on errors), all correct entries processed after the erroneous entry will be successfully added or modified.
- If a required attribute is not present, the modify operation fails. This happens even if the offending object class or attribute is not being modified. This situation can occur if you run the Directory Server with schema checking turned off, add unknown object classes or attributes, and then turn schema checking on.

For more information, see "Turning Schema Checking On and Off," on page 326.

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

Adding Entries Using ldapmodify

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

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

In this example, the LDIF statements in the **`new.ldif`** file do not specify a change type. They follow the format defined in "LDIF File Format," on page 471.

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

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

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

Parameter Name	Description
-a	Specifies that the modify operation will add new entries to the directory.

Parameter Name	Description
`-D`	Specifies the distinguished name with which to authenticate to the server. The value must be a DN recognized by the Directory Server, and it must also have the authority to modify the entries.
`-w`	Specifies the password associated with the distinguished name specified in the `-D` parameter.
`-h`	Specifies the name of the host on which the server is running.
`-p`	Specifies the port number that the server uses.
`-f`	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from `stdin`. For information on supplying LDIF update statements from the command line, refer to “Providing Input From the Command Line,” on page 47

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

Modifying Entries Using ldapmodify

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

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

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

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

The following table describes the `ldapmodify` 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 `-D` parameter.
`-h`	Specifies the name of the host on which the server is running.
`-p`	Specifies the port number that the server uses.
`-f`	Optional parameter that specifies the file containing the LDIF update statements used to define the modifications. If you do not supply this parameter, the update statements are read from `stdin`. For information on supplying LDIF update statements from the command line, refer to "Providing Input From the Command Line," on page 47.

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

Deleting Entries Using ldapdelete

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

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

For example, of the following three entries:

```
ou=People,dc=siroe,dc=com
cn=Paula Simon,ou=People,dc=siroe,dc=com
cn=Jerry O’Connor,ou=People,dc=siroe,dc=com
```

you can delete only the last two entries. The entry that identifies the People subtree can be deleted only if there aren't any entries below it. If you want to delete `ou=People,dc=siroe,dc=com`, you must first delete Paula Simon and Jerry O'Connor's entries, and all other entries in that subtree.

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

- You want to delete the entries identified by the distinguished names: `cn=Robert Jenkins,ou=People,dc=siroe,dc=com` and `cn=Lisa Jangles, ou=People,dc=siroe,dc=com`.
- You have created a database administrator that has the authority to modify the entries, and whose distinguished name is **`cn=Directory Manager, dc=siroe,dc=com`**.
- The database administrator's password is **`King-Pin`**.
- The server is located on **`cyclops`**.
- The server uses port number **`845`**.

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

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

The following table 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 `-D` parameter.
`-h`	Specifies the name of the host on which the server is running.
`-p`	Specifies the port number that the server uses.

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

Using Special Characters

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

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

Depending on the command-line utility you use, you should use either single or double quotation marks for this purpose. Refer to your operating system documentation for more information.

In addition, if you are using DNs that contain commas, you must escape the commas with a backslash (\). For example:

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

To delete user Patricia Fuentes from the Siroe Bolivia, S.A. tree, you would enter the following command:

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

LDIF Update Statements

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

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

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

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

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

The general format of LDIF update statements is as follows:

`dn:` *distinguished_name*
changetype_identifier
change_operation_identifier
list_of_attributes

-

change_operation_identifier
list_of_attributes
-

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

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

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

The following sections describe the change types in detail.

Adding an Entry Using LDIF

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

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

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

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

dn: cn=Pete Minsky,ou=People,dc=siroe,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=siroe,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=siroe,dc=com
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: Groups
```

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

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

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

Renaming an Entry Using LDIF

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

```
cn=Barry Nixon,ou=People,dc=siroe,dc=com
```

is:

```
cn=Barry Nixon
```

And the RDN for:

```
ou=People,dc=siroe,dc=com
```

is:

```
ou=People
```

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

For example, the entry

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

can be modified to be:

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

but it cannot be modified to be:

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

The following example can be used to rename Sue Jacobs to Susan Jacobs:

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

Because `deleteoldrdn` is `0`, this example retains the existing RDN as a value in the new entry. The resulting entry would therefore have a common name (cn) attribute set to both Sue Jacobs and Susan Jacobs in addition to all the other attributes included in the original entry. However, if you used

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

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

A Note on Renaming Entries

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

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

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

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

Modifying an Entry Using LDIF

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

- `add:` *attribute*

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

- `replace:` *attribute*

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

- `delete:` *attribute*

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

This section contains the following topics:

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

Adding Attributes to Existing Entries Using LDIF

You use `changetype:modify` with the add operation to add an attribute and an attribute value to an entry.

For example, the following LDIF update statement adds a telephone number to the entry:

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

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

```
dn: cn=Barney Fife,ou=People,dc=siroe,dc=com
changetype: modify
add: jpegphoto
jpegphoto: /path/to/photo
```

Alternatively, you can add a jpeg photograph to the directory using the following standard LDIF notation:

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

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

```
version:1
```

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

```
prompt% ldapmodify -D userDN -w user_passwd
```

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

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

Changing an Attribute Value Using LDIF

Use `changetype:modify` with the replace operation to change all values of an attribute in an entry.

For example, the following LDIF update statement changes Barney's manager from Sally Nixon to Wally Hensford:

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

If the entry has multiple instances of the attribute, then to change one of the attribute values, you must delete the attribute value that you want to change, and then add the replacement value. For example, consider the following entry:

```
cn=Barney Fife,ou=People,dc=siroe,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-1212
telephonenumber: 555-5678
```

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

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

Barney’s entry is now as follows:

```
cn=Barney Fife,ou=People,dc=siroe,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-5678
telephonenumber: 555-4321
```

Deleting All Values of an Attribute Using LDIF

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

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

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

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

Deleting a Specific Attribute Value Using LDIF

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

For example, consider the following entry:

```
cn=Barney Fife,ou=People,dc=siroe,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-1212
telephonenumber: 555-5678
```

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

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

Barney’s entry then becomes:

```
cn=Barney Fife,ou=People,dc=siroe,dc=com
objectClass: inetOrgPerson
cn: Barney Fife
sn: Fife
telephonenumber: 555-5678
```

Deleting an Entry Using LDIF

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

For example, of the following three entries:

```
ou=People,dc=siroe,dc=com
cn=Paula Simon,ou=People,dc=siroe,dc=com
cn=Jerry O’Connor,ou=People,dc=siroe,dc=com
```

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

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

```
dn: cn=Pete Minsky,ou=People,dc=siroe,dc=com
changetype: delete

dn: cn=Sue Jacobs,ou=People,dc=siroe,dc=com
changetype: delete
```

CAUTION Do not delete the suffix `o=NetscapeRoot`. The iPlanet Administration Server uses this suffix to store information about installed iPlanet Servers. Deleting this suffix could force you to reinstall all of your iPlanet servers, including the directory server.

Modifying an Entry in an Internationalized Directory

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

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

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

Maintaining Referential Integrity

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

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

How Referential Integrity Works

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

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

By default, when the referential integrity plug-in is enabled it performs integrity updates on the `member`, `uniquemember`, `owner`, and `seeAlso` attributes immediately after a delete or rename operation. You can, however, configure the behavior of the referential integrity plug-in to suit your own needs. You can:

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

Using Referential Integrity with Replication

There are certain limitations associated with the use of the referential integrity plug-in in a replication environment:

- You should never enable it on a dedicated consumer server (a server that contains only read-only replicas).
- You should never enable it on a server that contains a combination of read-write and read-only replicas.
- You can enable it on a master server that contains only read-write replicas.
- In the context of multi-master replication, you should enable it on just one master.

Configuring the Supplier Server

When your replication environment satisfies the conditions listed above, you can enable the referential integrity plug-in.

1. Enable the referential integrity plug-in.

   This task is described in “Enabling / Disabling Referential Integrity,” on page 66.

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

   This task is described in “Recording Updates in the Change Log,” on page 66.

3. Ensure that the referential integrity plug-in is disabled on all consumer servers.

NOTE Because the supplier server sends any changes made by the referential integrity plug-in to consumer servers, it is unnecessary to run the referential integrity plug-in on consumer servers.

Enabling/Disabling Referential Integrity

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

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to "Using the Directory Server Console," on page 26.

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

   The settings for the plugin are displayed in the right pane.

3. Check the Enable plugin check box to enable the plugin, clear it to disable it.
4. Click Save to save your changes.
5. For your changes to be taken into account, go to the Tasks tab, and select Restart the Directory Server.

Recording Updates in the Change Log

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

You can make this change from the Directory Server Console.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to “Using the Directory Server Console,” on page 26.

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

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

3. In the arguments list, replace the `referint` filename with the absolute path to the change log directory.

4. Click Save to save your changes.

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

Modifying the Update Interval

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

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

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

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to “Using the Directory Server Console,” on page 26.

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

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

3. In the arguments list, replace the value in the first text box by the appropriate time interval.

4. Click Save to save your changes.

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

Modifying the Attribute List

By default, the referential integrity is set up to update the `member`, `uniquemember`, `owner`, and `seeAlso` attributes. You can add or delete attributes to be updated from the Directory Server Console.

From the Directory Server Console

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

   For information on starting the Directory Server Console, refer to “Using the Directory Server Console,” on page 26.

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

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

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

4. Click Save to save your changes.

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

NOTE For best performance, the attributes set for updating should also be indexed. For information on indexing, see Chapter 8, “Managing Indexes.”

Chapter 3

Configuring Directory Databases

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

This chapter includes the following sections:

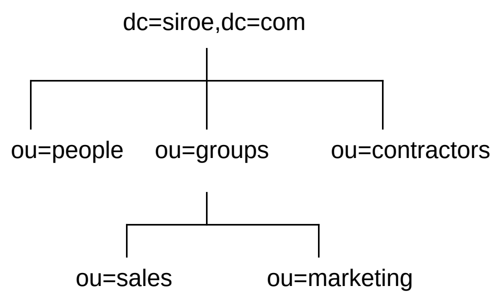
- Creating and Maintaining Suffixes
- Creating and Maintaining Databases
- Creating and Maintaining Database Links
- Using Referrals

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

Creating and Maintaining Suffixes

You can store different pieces of your directory tree in different databases, and then distribute these databases across multiple servers. Your directory tree contains branch points called nodes. These nodes can either be associated with databases or not. You create nodes using the Directory tab in the Directory Server Console, where you freely edit the entries that appear in your directory tree.

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



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

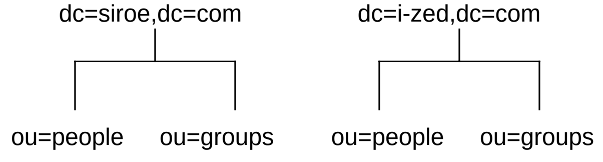
This section describes creating suffixes on your directory server and associating them with databases. This section contains procedures for the following:

- "Creating Suffixes," on page 72
- "Maintaining Suffixes," on page 78

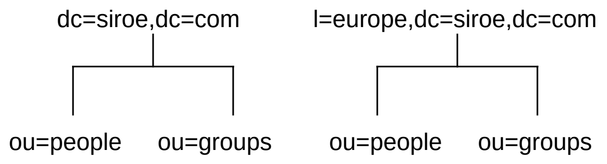
Creating Suffixes

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

Your directory might contain more than one root suffix. For example, an ISP might host several websites, one for siroe.com and one for i-zed.com. The ISP would create two root suffixes, one corresponding to the `dc=siroe,dc=com` naming context and one corresponding to the `dc=i-zed,dc=com` naming context. The directory tree appears as follows:

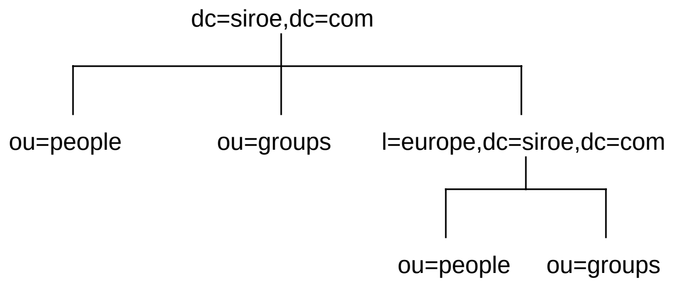


You can also create root suffixes to exclude portions of your directory tree from search operations. For example, Siroe Corporation might want to exclude their European office from a search on the general Siroe Corporation directory. To do this, they create two root suffixes. One root suffix corresponds to the general Siroe Corporation directory tree, `dc=siroe,dc=com`, and one root suffix corresponds to the European branch of their directory tree, `l=europe,dc=siroe,dc=com`. From a client application's perspective, the directory tree looks as follows:



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

If Siroe Corporation decides that they want to include the entries in the European branch of their directory tree in general searches, they would make the European branch a sub suffix of the general branch. To do this, they create a root suffix Siroe Corporation, `dc=siroe,dc=com`, and then create a sub suffix beneath it for their European directory entries, `l=europe,dc=siroe,dc=com`. From a client application's perspective, the directory tree would appear as follows:



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

- "Creating a New Root Suffix Using the Console," on page 74
- "Creating a New Sub Suffix Using the Console," on page 74
- "Creating Root and Sub Suffixes From the Command Line," on page 75

Creating a New Root Suffix Using the Console

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

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

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

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

   Deselect the checkbox if you want to create a database for the new root suffix later. You may want to do this if you want to be able to specify a directory where the database will be created. The new root suffix will be disabled until you create a database.
5. If you selected the "Create associated database automatically" checkbox in step 4, enter a unique name for the new database in the "Database name" field.

   Use only ASCII (7-bit) characters for naming the database. This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote ('), double quote ("), or a question mark (?). For example, you might name the new database `siroe2`.
6. Click OK to create the new root suffix.

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

Creating a New Sub Suffix Using the Console

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

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

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

3. Enter a unique suffix name in the "New suffix" field.

   The suffix must be named according to dc naming conventions. For example, you might enter a new suffix name of `ou=groups`.

   The root suffix is automatically added to the name. For example, if you are creating the sub suffix `ou=groups` under the `dc=siroe,dc=com` suffix, the console automatically names it `ou=groups,dc=siroe,dc=com`.

4. Select the "Create associated database automatically" checkbox if you want a database to be created at the same time as the new sub suffix.

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

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

   Use only ASCII (7-bit) characters for naming the database. This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote ('), double quote ("), or a question mark (?). For example, you might name the new database `siroe3`.

6. Click OK to create the new sub suffix.

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

Creating Root and Sub Suffixes From the Command Line

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

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

```
cd /usr/iplanet/servers/shared/bin
```

Then, run `ldapmodify` as follows:

```
ldapmodify -a -h siroe1 -p 389 -D "cn=directory manager" -w secret
```

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

Next, you create the root suffix entry for Siroe Corporation as follows:

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

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

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

NOTE If you want to maintain your suffixes using the Directory Server Console, you will need to respect the same spacing you use to name the root and sub suffixes via the command line.

For example, if you name a root suffix `ou=groups  ,dc=siroe,dc=com` (with two spaces after `groups`), any sub suffixes you create under this root will need to specify two spaces after `ou=groups` as well.

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

Table 3-1 Suffix Attributes

Attribute Name	Value
dn	Defines the DN for the suffix. The DN is contained in quotes. The value you enter takes the following form: `cn=”dc=domain,dc=com”,cn=mapping tree, cn=config` This attribute is required.
cn	Defines the relative DN (RDN) of the entry. This attribute is required.

Table 3-1 Suffix Attributes

Attribute Name	Value
objectclass	Tells the server that the entry is root or sub suffix entry. It always takes the value `nsMappingTree`. This attribute is required.
`nsslapd-state`	Determines how the suffix handles operations. This attribute takes the following values: • `backend`: the backend (database) is used to process all operations. • `disabled`: the database is not available for processing operations. The server returns a “No such search object” error in response to requests made by client applications. • `referral`: a referral is returned for requests made to this suffix. • `referral on update`: the database is used for all operations except update requests, which receive a referral. The default value is `disabled`.
`nsslapd-referral`	Defines the LDAP URL of the referral to be returned by the suffix. This attribute can be multi valued, with one referral per value. This attribute is required when the value of the `nsslapd-state` attribute is `referral` or `referral on update.`
`nsslapd-backend`	Gives the name of the database or database link used to process requests. This attribute can be multi valued, with one database or database link per value. Refer to “Creating and Maintaining Database Links,” on page 87 for more information about database links. This attribute is required when the value of the `nsslapd-state` attribute is set to `backend` or `referral on update`.
`nsslapd-distribution-plugin`	Specifies the shared library to be used with the custom distribution function. This attribute is required only when you have specified more than one database in the `nsslapd-backend` attribute. Refer to “Creating and Maintaining Databases,” on page 81 for more information about the custom distribution function.

Table 3-1 Suffix Attributes

Attribute Name	Value
`nsslapd-distribution-funct`	Specifies the name of your custom distribution function. This attribute is required only when you specify more than one database in the `nsslapd-backend` attribute. Refer to "Creating and Maintaining Databases," on page 81 for more information about the custom distribution function.
`nsslapd-parent-suffix`	Provides the DN of the parent entry for a sub suffix. By default, this attribute is not present, which means that the suffix is regarded as a root suffix. For example, you want to create a sub suffix `o=sales,dc=siroe,dc=com` under the root suffix `dc=siroe,dc=com`. Add the following value to the `nsslapd-parent-suffix` attribute of the sub suffix: `nsslapd-parent-suffix: "dc=siroe,dc=com"`

Maintaining Suffixes

This section describes the following procedures:

- "Using Referrals in a Suffix," on page 78
- "Enabling Referrals Only During Update Operations," on page 79
- "Disabling a Suffix," on page 80
- "Deleting a Suffix," on page 80

Using Referrals in a Suffix

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

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

To set referrals in a suffix:

1. On the Directory Server Console, select the Configuration tab.
2. Under Data in the left pane, click the suffix to which you want to add a referral.

3. Click the Suffix Settings tab. Select the Use Referrals radio button.
4. Click the Referrals tab. Enter an LDAP URL in the “Enter a new referral” field or click Construct to be guided through the creation of an LDAP URL.

   For more information about the structure of LDAP URLs, refer to Appendix C, “LDAP URLs.”
5. Click Add to add the referral to the list.

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

Enabling Referrals Only During Update Operations

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

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

To enable referrals only during update operations:

1. On the Directory Server Console select the Configuration tab.
2. Under Data in the left pane, click the suffix to which you want to add a referral.
3. Click the Suffix Settings tab. Select the “Use Referrals on Update” radio button.
4. Click the Referrals tab. Enter an LDAP URL in the “Enter a new referral” field or click Construct to be guided through the creation of an LDAP URL.

   For more information about the structure of LDAP URLs, refer to Appendix C, “LDAP URLs.”
5. Click Add to add the referral to the list.

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

Disabling a Suffix

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

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

To disable a suffix:

1. On the Directory Server Console select the Configuration tab.
2. Under Data in the left navigation pane, click the suffix you want to disable.
3. Click the Suffix Setting tab. Deselect the “Enable this suffix” checkbox.

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

   The suffix is no longer enabled.

Deleting a Suffix

The following procedure describes deleting a suffix:

CAUTION When you delete a suffix, you also delete all database entries and replication information associated with that suffix.

1. On the Directory Server Console select the Configuration tab.
2. Under Data in the left navigation pane, select the suffix you want to delete.
3. Select “Delete...” from the Object menu.

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

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

5. Click OK to delete the suffix.

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

Creating and Maintaining Databases

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

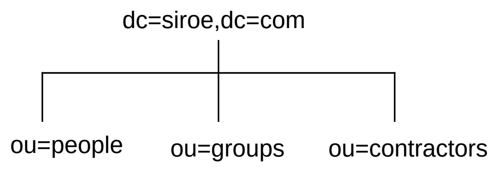
This section contains information about creating databases to contain your directory data, deleting databases, and making databases temporarily read-only.

Creating Databases

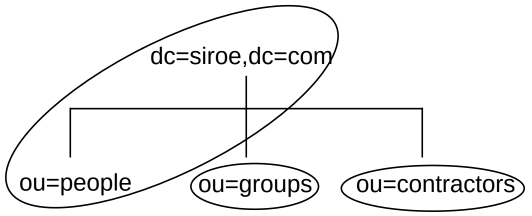
iPlanet Directory Server 5.0 supports the use of multiple databases over which you can distribute your directory tree. There are two ways you can distribute your data across multiple databases:

- One database per suffix.

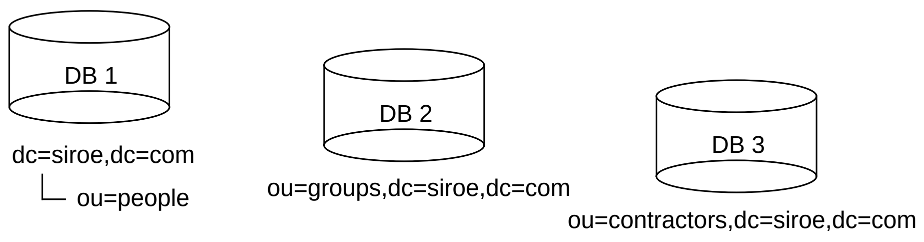
  The data for each suffix is contained in a separate database. For example, your directory tree appears as follows:



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



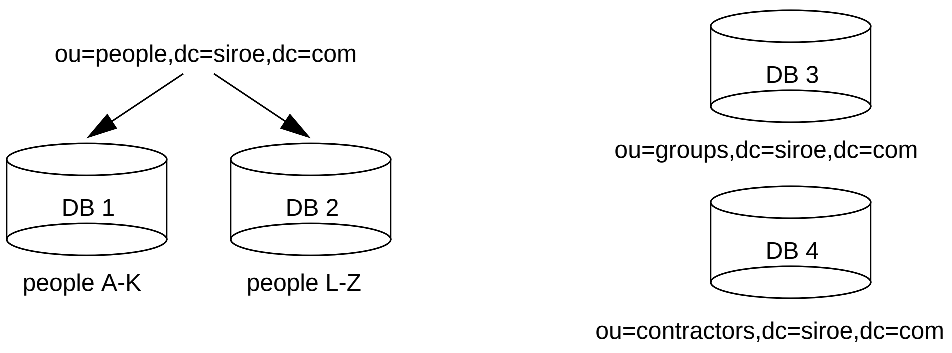
This division of the tree corresponds to three databases as follows:



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

- Multiple databases for one suffix.

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


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

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

You need to use the custom distribution plug-in to distribute data from a single suffix across multiple databases. Contact iPlanet Professional Services for information on how to create distribution logic for your directory server. For more information about Professional Services, go to http://www.iplanet.com/services/l.

Creating a New Database for an Existing Suffix Using the Console

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

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

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

   This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote (‘), double quote (“), or a question mark (?). For example, you might name the new database `siroe2`.
5. In the “Create database in” field, enter the path to the directory where you want to store the new database. You can also click Browse to locate a directory on your local machine.

   By default, the directory stores the new database in the `/usr/iplanet/servers/slapd-`*serverID*`/db` directory.
6. Click OK. Click Yes in the confirmation dialog to create the new database.

NOTE	To see the new suffix in the Directory tab, you first need to create a root entry associated with the suffix. Refer to “Creating Directory Entries,” on page 39.

Creating a New Database for a Single Suffix From the Command Line

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

For example, you want to add a new database to the server siroe1. First, type the following to change to the directory containing the `ldapmodify` utility:

```
cd /usr/iplanet/servers/shared/bin
```

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

```
ldapmodify -a -h siroe1 -p 389 -D "cn=directory manager" -w secret
```

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

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

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

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

To create a root or sub suffix from the command line, refer to "Creating Root and Sub Suffixes From the Command Line," on page 75. The database name, given in the DN attribute, must correspond with the value in the `nsslapd-backend` attribute of the suffix entry.

Adding Multiple Databases for a Single Suffix

You can distribute a single suffix across multiple databases. However, to distribute the suffix you need to create a custom distribution function to extend the directory. For more information on creating a custom distribution function, contact iPlanet Professional Services. For more information about Professional Services, go to http://www.iplanet.com/services/l.

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

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

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

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

Adding the Custom Distribution Function to a Suffix

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

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

Adding Custom Distribution Using the Console

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

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

Adding Custom Distribution From the Command Line

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

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

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

For more information about using the `ldapmodify` command-line utility, refer to "Adding and Modifying Entries Using ldapmodify," on page 49.

Maintaining Directory Databases

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

- "Placing a Database in Read-Only Mode," on page 86
- "Deleting a Database," on page 87

Placing a Database in Read-Only Mode

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

If your directory server manages multiple databases, you can place all of them into read-only mode at the same time by placing your entire server in read-only mode. For more information, see "Placing the Entire Directory Server in Read-Only Mode," on page 32.

This section includes procedures for the following:

- "Making a Database Read-Only Using the Console," on page 86
- "Making a Database Read-Only From the Command Line," on page 87

Making a Database Read-Only Using the Console

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

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

Making a Database Read-Only From the Command Line

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

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

Deleting a Database

The following procedure describes deleting a directory database using the Directory Server Console. Deleting a database deletes the configuration information and entries for that database only, not the physical database itself.

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

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

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

   A progress dialog box appears telling you the steps the directory server completes during the deletion.

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

Creating and Maintaining Database Links

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

The following sections describe how to create and configure a database link. For more general information about chaining, refer to “Designing the Directory Topology” in *iPlanet Directory Server Deployment Guide*.

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

- “Configuring the Chaining Policy,” on page 88
- “Creating a New Database Link,” on page 94
- “Chaining Using SSL,” on page 104
- “Maintaining Database Links,” on page 105
- “Database Links and Access Control Evaluation,” on page 106
- “Advanced Feature: Tuning Database Link Performance,” on page 108
- “Advanced Feature: Configuring Cascading Chaining,” on page 113

For information about monitoring the activity of your database links, refer to “Monitoring Database Link Activity,” on page 397.

Configuring the Chaining Policy

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

This section contains the following information:

- “Chaining Component Operations,” on page 88
- “Chaining LDAP Controls,” on page 92

Chaining Component Operations

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

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

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

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

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

Table 3-2 Components Allowed to Chain

Component Name	Description	Permissions
ACI plug-in	This plug-in implements the access control feature. Operations used to retrieve and update ACI attributes are not chained because it is not safe to mix local and remote ACI attributes. However, requests used to retrieve user entries may be chained. Specify the following value in `nsActiveChainingComponents` attribute: `nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config`	Read, search, and compare
4.0 plug-ins	This component name represents all Directory Server 4.0 plug-ins. The 4.0 plug-ins share the same chaining policy. Specify the following in the `nsActiveChainingComponents` attribute: `nsActiveChainingComponents: cn=old plugin,cn=plugins,cn=config`	Depends upon the 4.0 plug-in you are allowing to chain
Resource limit component	This component sets server limits depending on the user bind DN. You can apply resource limits on remote users if the resource limitation component is allowed to chain. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=resource limits,cn=components,cn=config`	Read, search, and compare

Table 3-2 Components Allowed to Chain

Component Name	Description	Permissions
Certificate-based authentication checking component	This component is used when the SASL-external bind method is used. It retrieves the user certificate from the database on the remote server. If you allow this component to chain, certificate-based authentication can work with a database link. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=certificate-based authentication,cn=components,cn=config`	Read, search, and compare
Referential integrity plug-in	This plug-in ensures that updates made to attributes containing DNs are propagated to all entries that contain pointers to the attribute. For example, if you delete an entry that is a member of a group, the entry is automatically removed from the group. Using this plug-in with chaining helps simplify the management of static groups when the group members are remote to the static group definition. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=referential integrity postoperation,cn=plugins,cn=config`	Read, write, search, and compare
UID uniqueness plug-in	This plug-in checks that all the values for a specified `uid` attribute are unique (no duplicates). If you allow this plug-in to chain, it confirms that the `uid` attribute values are unique even on attributes changed through a database link. To chain this component's operations, specify the following: `nsActiveChainingComponents: cn=uid uniqueness,cn=plugins,cn=config`	Read, search, and compare

You cannot chain the following components:

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

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

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

Chaining Component Operations Using the Console

1. On 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. After making modifications to the components list, a red dot appears on the tab and the field name turns gray. Click Save to save your changes.

   Restart the server in order for the change to take affect.

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

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

Chaining Component Operations From the Command Line

You can specify components you want to include in chaining using the `nsActiveChainingComponents` attribute in the `cn=config,cn=chaining database,cn=plugins,cn=config` entry of the configuration file.

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

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

Refer to Table 3-2 on page 89 for a list of the components you can allow to chain.

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

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

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

Chaining LDAP Controls

You can choose to not chain operation requests made by LDAP controls. By default, requests made by the following controls are forwarded to the remote server by the database link:

- Virtual list view (VLV).

  This control provides lists of parts of entries rather than returning all entry information.

- Server side sorting.

  This control sorts entries according to their attribute values.

- Managed DSA.

  This controls returns smart referrals as entries rather than following the referral. This allows you to change or delete the smart referral itself.

- Loop detection.

  This control keeps track of the number of times the server chains with another server. When the count reaches a number you configure, a loop is detected and the client application is notified.

  For more information about using this control, refer to “Detecting Loops,” on page 120.

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

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

Chaining LDAP Controls Using the Console

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

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

Chaining LDAP Controls From the Command Line

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

```
nsTransmittedControls: 2.16.840.1.113730.3.4.9
```

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

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

Table 3-3 LDAP Controls and Their OIDs

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

For more information about LDAP controls, refer to the LDAP C-SDK documentation on http://docs.iplanet.com/docs/manuals/directory.html.

Creating a New Database Link

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

Suffix information. You create a suffix in your directory tree that is managed by the database link, not a regular database. This suffix corresponds to the suffix on the remote server that contains the data.

Bind credentials. When the database link binds to a remote server, it impersonates a user. You need to specify the DN and the credentials you want each database link to use to bind with remote servers.

LDAP URL. You provide the LDAP URL of the remote server to which the database link connects.

List of failover servers. You can provide a list of alternative servers for the database link to contact in the event of a failure. This configuration item is optional.

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

Creating a New Database Link Using the Console

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

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

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

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

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

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

6. In the left pane, right-click the suffix you just created and select “New Database Link” from the pop-up menu.

   The Create New Database Link dialog box is displayed.

7. Enter the name of the new database link in the “Database link name” field.

   Use only ASCII (7-bit) characters for naming the database link. This value cannot contain commas, tabs, an equals sign (=), asterisk (*), backslash (\), forward slash (/), plus sign (+), quote (‘), double quote (“), or a question mark (?). For example, you might name the new database link `siroelink1`.

8. Enter the DN used by the database link to bind to the remote server in the “Bind DN” field.

   For example, you might enter `cn=dblink` in the “Bind DN” field.

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

10. Select the “Use a secure LDAP connection between servers” checkbox if you want the database link to use SSL to communicate to the remote server.

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

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

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

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

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

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

Creating a Database Link From the Command Line

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

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

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

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

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

- "Providing Suffix Information," on page 96
- "Providing Bind Credentials," on page 97
- "Providing an LDAP URL," on page 99
- "Providing a List of Failover Servers," on page 99
- "Summary of Cascading Chaining Configuration Attributes," on page 120
- "Database Link Configuration Example," on page 102

Providing Suffix Information

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

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

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

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

Providing Bind Credentials

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

Providing bind credentials involves the following steps:

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

   a. Create an administrative user for the database link.

      For information on adding entries, see “Creating Directory Entries,” on page 37.

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

      For more information on configuring ACI’s, refer to “Managing Access Control,” on page 183.

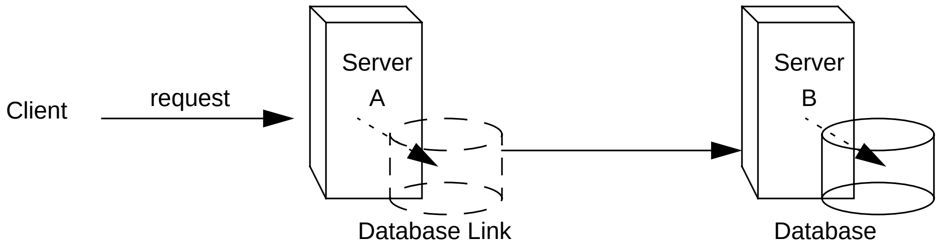
2. On the server containing the database like, you need to do the following:

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

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

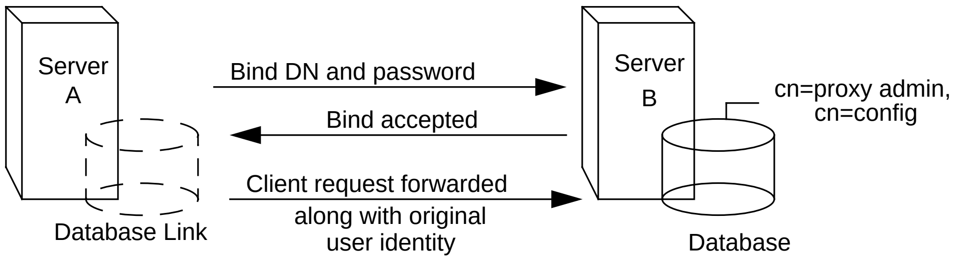
   b. Use `ldapmodify` to provide a user password for the database link in the `nsMultiplexorCredentials` attribute of the `cn=`*database_link_name*`,cn=chaining database,cn=plugins,cn=config` entry.

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



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

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


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

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

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

For more information on ACIs, refer to "Managing Access Control," on page 183. For more information about the proxy authentication control, refer to the C-SDK documentation at http://developer.iplanet.com/docs/manuals/directory.html.

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

Providing an LDAP URL

On the server containing your database link, you have to identify the remote server that the database link connects with using an LDAP URL. Unlike the standard LDAP URL format, the URL of the remote server does not specify a suffix. It takes the following form:

`ldap://`*servername:portnumber*`/`

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

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

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

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

`ldaps://`*servername:portnumber*`/`

For more information about chaining and SSL, refer to "Chaining Using SSL," on page 104.

Providing a List of Failover Servers

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

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

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

Summary of Database Link Configuration Attributes

The following table lists the attributes available for configuring a database link. Some of these attributes were discussed in the earlier sections.

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

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

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

Table 3-4 Database Link Configuration Attributes

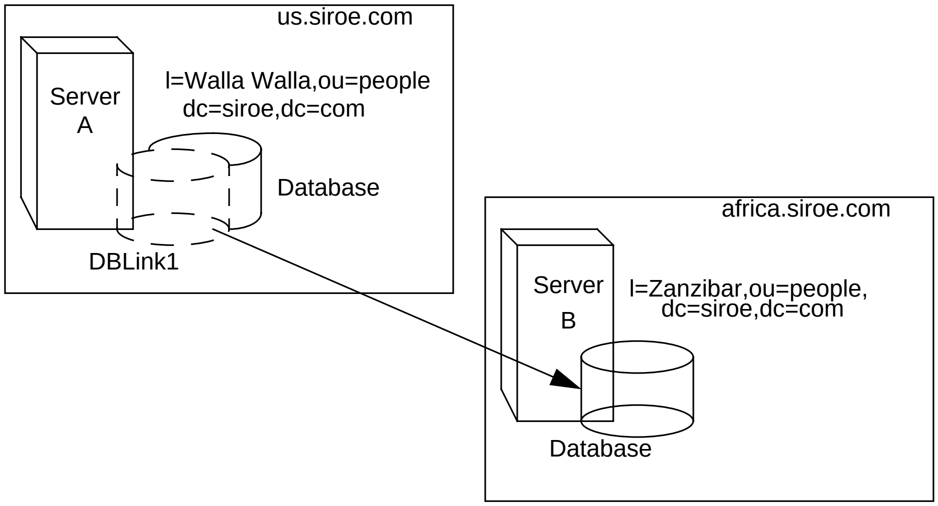
Attributes	Value
`*nsTransmittedControls`	Gives the OID of LDAP controls forwarded by the database link to the remote data server.
`nsslapd-suffix`	The suffix managed by the database link. Any changes you make to this attribute after the entry has been created take effect only after you restart the server containing the database link.
`nsslapd-timelimit`	Default search time limit for the database link, given in seconds. The default value is 3600 seconds.
`nsslapd-sizelimit`	Default size limit for the database link, given in number of entries. The default value is 2000 entries.
`nsFarmServerURL`	Gives the LDAP URL of the remote server (or farm server) that contains the data. This attribute can contain optional servers for failover, separated by spaces. If using cascading chaining, this URL can point to another database link.

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

Attributes	Value
`nsMultiplexorBindDN`	DN of the administrative entry used to communicate with the remote server. The term *multiplexor* in the name of the attribute means the server which contains the database link and communicates with the remote server. This bind DN cannot be the directory manager. If this attribute is not specified, the database link binds as anonymous.
`nsMultiplexorCredentials`	Password for the administrative user, given in plain text. If no password is provided, it means that users can bind as anonymous. The password is encrypted in the configuration file.
`nsCheckLocalACI`	Reserved for advanced use only. Controls whether ACIs are evaluated on the database link as well as the remote data server. Takes the values `on` or `off`. Changes to this attribute occur only after the server has been restarted. The default value is `off`.
`nsProxiedAuthorization`	Reserved for advanced use only. Allows you to disable proxied authorization. A value of `off` means proxied authorization is disabled. The default value is `on`.
`*nsActiveChainingComponents`	Lists the components using chaining. A component is any functional unit in the server. The value of this attribute in the database link instance overrides the value in the global configuration attribute. To disable chaining on a particular database instance, use the value `none`. The default policy is to not allow chaining. Refer to “Chaining Component Operations,” on page 88 for more information.
`nsReferralOnScopedSearch`	Controls whether or not referrals are returned by scoped searches. This attribute is for optimizing your directory, because returning referrals in response to scoped searches is more efficient. Takes the values `on` or `off`. The default value is `off`.
`nsHopLimit`	Maximum number of times a request can be forwarded from one database link to another. The default value is `10`.

Database Link Configuration Example

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


us.siroe.com
Server A
l=Walla Walla,ou=people dc=siroe,dc=com
Database
DBLink1
africa.siroe.com
Server B
l=Zanzibar,ou=people, dc=siroe,dc=com
Database

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

```
cd /usr/iplanet/servers/shared/bin
```

Run the script as follows:

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

Then specify the configuration information for the database link:

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

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

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

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

Next, you create an administrative user on server B as follows:

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

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

Add the following proxy authorization ACI to the `l=Zanzibar,ou=people,dc=siroe,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=siroe,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 successfully modify or write data to the remote server, you need to set up the correct access controls on the remote server.

For more information about how access controls are evaluated in the context of chained operations, refer to "Database Links and Access Control Evaluation," on page 106.

Chaining Using SSL

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

- Enable SSL on the remote server.

  For more information on enabling SSL, refer to "Enabling SSL: Summary of Steps," on page 362.

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

  You specify the LDAP URL in the `nsFarmServerURL` attribute. For more information about this attribute, see "Providing an LDAP URL," on page 99.

  For example, you might specify the following LDAP URL:

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

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

  For more information on enabling SSL, refer to "Enabling SSL: Summary of Steps," on page 362.

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

Maintaining Database Links

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

- "Updating Remote Server Authentication Information," on page 105
- "Deleting Database Links," on page 106

Updating Remote Server Authentication Information

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

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

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

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

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

Deleting Database Links

To delete a database link:

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

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

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

   A progress dialog box appears telling you the steps the directory server completes during the deletion.

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

Database Links and Access Control Evaluation

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

- You cannot use all types of access control.

  For example, role based or filter based ACIs need access to the user entry. Because you are accessing the data via database links, only the data in the proxy control can be verified. Consider designing your directory in a way that ensures the user entry is located in the same database as the user’s data.
- All access controls based on the IP address or DNS domain of the client may not work, as the original domain of the client is lost during chaining.

  The remote server views the client application as being at the same IP address and in the same DNS domain as the database link.

The following restrictions apply to the ACIs you create to use with database links:

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

Though access controls are always evaluated on the remote server, you can also choose to have them evaluated on both the server containing the database link and the remote server. This poses several limitations:

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

  For performance reasons, clients cannot do remote inquiries and evaluate access controls.
- The database link does not necessarily have access to the entries being modified by the client application.

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

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

Advanced Feature: Tuning Database Link Performance

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

- "Managing Connections to the Remote Server," on page 108
- "Detecting Errors During Normal Processing," on page 111
- "Managing Threaded Operations," on page 112

Managing Connections to the Remote Server

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

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

Managing Connections to the Remote Server Using the Console

1. On the Directory Server Console, select the Configuration tab.
2. Expand the Data folder in the left pane and locate the database link you want to change. Click the database link, then click the Limits and Controls tab in the right navigation pane.
3. In the Connection Management section, make changes to any of the following fields:

   Maximum TCP connection(s). The maximum number of TCP connections that the database link establishes with the remote server. The default value is `3` connections.

   Bind timeout. Amount of time, in seconds, before the database link's bind attempt times out. The default value is `15` seconds.

   Maximum binds per connection. Maximum number of outstanding bind operations per TCP connection. The default value is `10` outstanding bind operations per connection.

   Time out before abandon (sec). Number of seconds before the server checks to see if a timed out connection should be abandoned. The default value is `2`seconds.

Maximum LDAP connection(s). Maximum number of LDAP connections that the database link establishes with the remote server. The default value is `10` connections.

Maximum bind retries. Number of times a database link attempts to bind to the remote server. A value of 0 indicates that the database link will try to bind only once. The default value is `3` attempts.

Maximum operations per connection. Maximum number of outstanding operations per LDAP connection. The default value is `10` operations per connection.

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

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

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

4. Click Save to save your changes.

Managing Connections to the Remote Server From the Command Line

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

The default connection management attributes are stored in the following entry: `cn=default instance config`, `cn=chaining database,cn=plugins,cn=config`.

The connection management attributes for a specific database link are stored in the following entry: `cn=`*database_link_name*,`cn=chaining database,cn=plugins,cn=config`, where *database_link_name* is the name of the database link. The connection management attributes specified in this entry take precedence over the attributes specified in the `cn=default instance config` entry.

The following table lists the attributes associated with connection management:

Table 3-5 Database Link Connection Management Attributes

Attribute Name	Description
nsOperationConnectionsLimit	Maximum number of LDAP connections that the database link establishes with the remote server. The default value is 10 connections per database link instance.
nsBindConnectionsLimit	Maximum number of TCP connections that the database link establishes with the remote server. The default value is 3 connections.
nsConcurrentOperationsLimit	Maximum number of outstanding operations per LDAP connection. The default value is 10 operations per connection.
nsConcurrentBindLimit	Maximum number of outstanding bind operations per TCP connection. The default value is 10 outstanding bind operations.
nsBindRetryLimit	Number of times a database link attempts to bind to the remote server. A value of zero (0) indicates that the database link will try to bind only once. The default value is 3 attempts.
nsConnectionLife	Connection lifetime, in seconds. You can keep connections between the database link and the remote server open for an unspecified time, or you can close them after a specific period of time. It is faster to keep the connections open, but it uses more resources. For example, if you are using a dial-up connection you may want to limit the connection time. A value of 0 indicates there is no limit. By default, the value is set to 0. When the value is 0 and you provide a list of failover servers in the nsFarmServerURL attribute, the “main” server is never contacted after failover to the alternate server. The default value is 0 seconds.
nsBindTimeout	Amount of time, in seconds, before the bind attempt times out. The default value is 15 seconds.
nsAbandonedSearchCheckInterval	Number of seconds that pass before the server checks for abandoned operations. The default value is 10 seconds.

For the list of database link configuration attributes, refer to “Database Link Configuration Attributes,” on page 100.

Detecting Errors During Normal Processing

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

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

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

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

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

Table 3-6 Database Link Processing Error Detection Parameters

Attribute Name	**Description**
nsMaxResponseDelay	Maximum amount of time it can take a remote server to respond to an LDAP operation request made by a database link before an error is suspected.This period is given in seconds. The default delay period is 60 seconds. Once this delay period has been met, the database link tests the connection with the remote server.

Table 3-6 Database Link Processing Error Detection Parameters

Attribute Name	Description
nsMaxTestResponseDelay	Duration of the test issued by the database link to check whether the remote server is responding. If a response from the remote server is not returned before this period has passed, the database link assumes the remote server is down and the connection is not used for subsequent operations. This period is given in seconds. The default test response delay period is 15 seconds.

Managing Threaded Operations

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

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

While the database link waits for results from the remote server, it can process additional operations. By default, the number of threads used by the server is 5. However, when using database links, you can improve performance by increasing the number of threads available for processing operations. While the local CPU waits for a response from a remote server, it can process other operations rather than stand idle.

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

Advanced Feature: Configuring Cascading Chaining

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

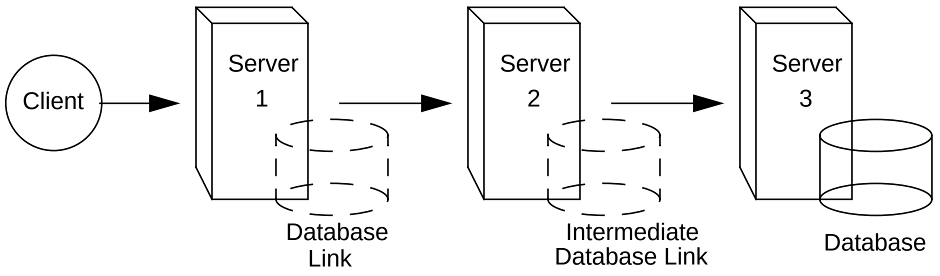
The section contains the following topics:

- "Overview of Cascading Chaining," on page 113
- "Configuring Cascading Chaining Defaults Using the Console," on page 115
- "Configuring Cascading Chaining Using the Console," on page 116
- "Configuring Cascading Chaining From the Command Line," on page 117
- "Summary of Cascading Chaining Configuration Attributes," on page 120
- "Cascading Chaining Configuration Example," on page 121

Overview of Cascading Chaining

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

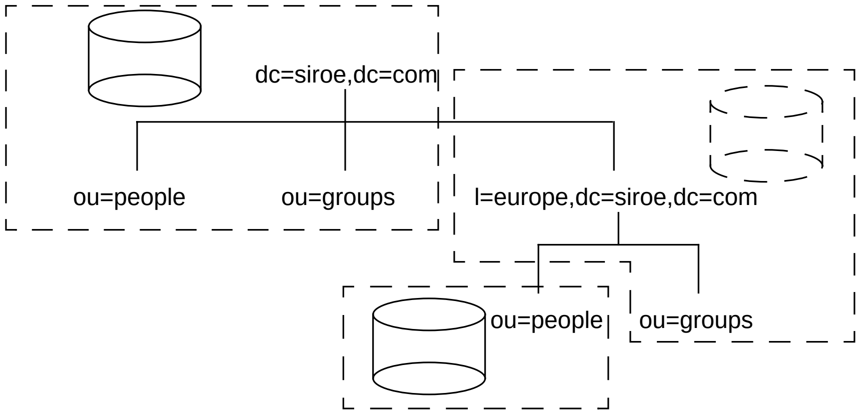
For example, consider the following scenario:



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

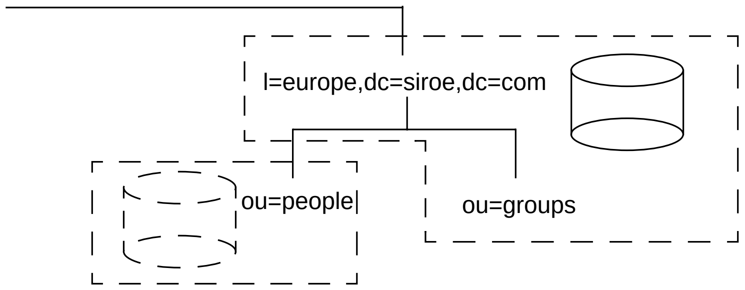
During a normal operation request, a client binds to the server and then any ACIs applying to that client are evaluated. With cascading chaining, the client bind request is evaluated on server one, but the ACIs applying to the client are evaluated only after the request has been chained to the destination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

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

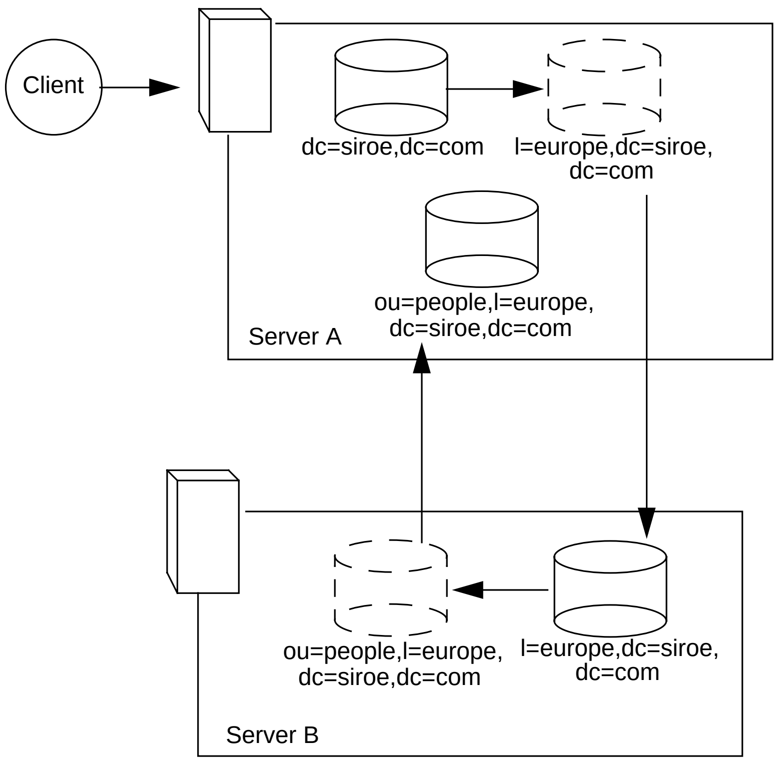


The root suffix and the `ou=people` and `ou=groups` sub suffixes are stored in one database. The `l=europe,dc=siroe,dc=com` and `ou=groups` suffixes are stored in a database on server B, which server A contacts using a database link. The `ou=people` branch of the `l=europe,dc=siroe,dc=com` suffix is stored in a second database on server A.

The portion of this directory tree on server B is stored as follows:



So, a client request targeted at the `ou=people,l=europe,dc=siroe,dc=com` entry would be routed by the directory as follows:



Because at least two hops are required for the directory to service the client request, this is considered a cascading chain. First server A chains to a database on server B, then server B chains back to server A to access the data in the `ou=people,l=europe,dc=siroe,dc=com` branch.

Configuring Cascading Chaining Defaults Using the Console

To set cascading chaining defaults for all database links in your directory server:

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

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

   This is an advanced feature. For more information, refer to "Enabling Local ACI Evaluation," on page 119.

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

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

5. Click Save to save your changes.

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

Configuring Cascading Chaining Using the Console

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

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

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

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

   This is an advanced feature. For more information, refer to "Enabling Local ACI Evaluation," on page 119.

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

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

5. Click Save to save your changes.

Configuring Cascading Chaining From the Command Line

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

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

This section covers the following topics:

- "Pointing to Another Database Link," on page 117
- "Transmitting the Proxy Authorization Control," on page 118
- "Creating the Proxy Administrative User ACI," on page 118
- "Enabling Local ACI Evaluation," on page 119
- "Creating Client ACIs," on page 119
- "Detecting Loops," on page 120

Pointing to Another Database Link

To create a cascading chain, the `nsFarmServerURL` attribute of one database link must contain the URL of the server containing another database link. For example, suppose the database link on the server called `siroe1.com` points to a database link on the server called `africa.siroe.com`. The `cn=`*database_link_name*`,cn=chaining database, cn=plugins,cn=config` entry of the database link on server one would contain the following:

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

Transmitting the Proxy Authorization Control

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

```
nsTransmittedControls: 2.16.840.1.113730.3.4.12
```

The OID value represents the Proxy Authorization Control. For more information about chaining LDAP controls, refer to “Chaining LDAP Controls,” on page 92.

Creating the Proxy Administrative User ACI

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

To prevent this security hole, you need to create an ACI. To create an ACI, you need to do the following:

1. Create a database, if one does not already exist, on the server containing the intermediate database link. This database will contain the admin user entry and the ACI. For information about creating a database, see “Creating Databases,” on page 81.
2. Create an entry that corresponds to the administrative user in the database.
3. Create an ACI for the administrative user that targets the appropriate suffix. This ensures the administrator has access only to the suffix of the database link. Add the following ACI to the administrative user’s entry:

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

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

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

Enabling Local ACI Evaluation

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

```
nsCheckLocalACI: on
```

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

Creating Client ACIs

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

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

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

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

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

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

Detecting Loops

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

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

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

```
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

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

Summary of Cascading Chaining Configuration Attributes

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

Table 3-7 Cascading Chaining Configuration Attributes

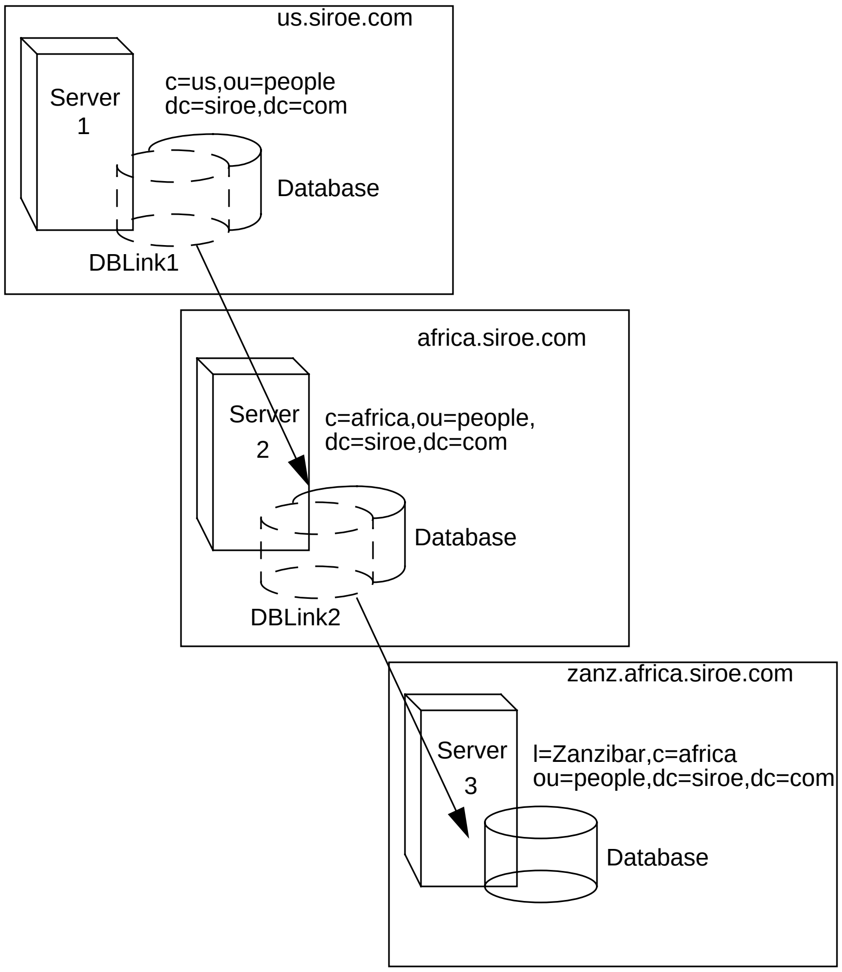
Attribute	Description
nsFarmServerURL	URL of the server containing the next database link in the cascading chain.

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

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

Cascading Chaining Configuration Example

For example, you create a cascading chain involving three servers as follows:



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

```
cd /usr/iplanet/servers/shared/bin
```

Run the utility as follows:

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

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

```
dn: cn=DBLink1,cn=chaining database,cn=plugins,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: l=Zanzibar,c=africa,ou=people,dc=siroe,dc=com
nsfarmserverurl: ldap://africa.siroe.com:389/
nsmultiplexorbinddn: cn=server2 proxy admin,cn=config
nsmultiplexorcredentials: secret
cn: DBLink1
```

```
cn="l=Zanzibar,c=africa,ou=people,dc=siroe,dc=com",cn=mapping
tree,cn=config
objectclass=nsMappingTree
nsslapd-state=backend
nsslapd-backend=DBLink1
nsslapd-suffix: l=Zanzibar,c=africa,ou=people,dc=siroe,dc=com
cn: l=Zanzibar,c=africa,ou=people,dc=siroe,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.

Next you add the following information to the `cn=DBLink1,cn=chaining database,cn=plugins,cn=config` entry on server one:

```
nsCheckLocalACI: on
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

This attribute enables the evaluation of local ACIs, and is necessary when chaining. You want to implement loop detection, so you specify the OID of the loop detection control in the `nsTransmittedControl` attribute.

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

```
dn: cn=DBLink2,cn=chaining database,cn=plugins,cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsBackendInstance
nsslapd-suffix: l=Zanzibar,c=africa,ou=people,dc=siroe,dc=com
nsfarmserverurl: ldap://zanz.africa.siroe.com:389/
nsmultiplexorbinddn: cn=server3 proxy admin,cn=config
nsmultiplexorcredentials: secret
cn: DBLink2
```

```
dn: cn="l=Zanzibar,c=africa,ou=people,dc=siroe,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=siroe,dc=com”
cn: l=Zanzibar,c=africa,ou=people,dc=siroe,dc=com
```

Next, you create an administrative user on server one. This administrative user will be used to bind and authenticate to server two. You create the proxy administrative user as follows:

```
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
```

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

Add the following proxy authorization ACI to the `l=Zanzibar, ou=people,dc=siroe,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 within the `l=Zanzibar,ou=people,dc=siroe,dc=com` subtree only.

The database link on server two must be configured to transmit the proxy authorization control and, if you want to implement loop detection, the loop detection control. To do this you add the following information to the `cn=DBLink2,cn=chaining database,cn=plugins,cn=config` entry of server two:

```
nsCheckLocalACI: on
nsTrasmittedControl: 2.16.840.1.113730.3.4.12
nsTransmittedControl: 1.3.6.1.4.1.1466.29539.12
```

The `nsCheckLocalACI` attribute enables the evaluation of local ACIs. The `nsTransmittedControl` attributes transmit both the proxy authorization control and the loop detection control.

Finally, you need to create an ACI for the client application making the request.To create this ACI, you need to first create a database on server two 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.

You create the following database to contain the ACI for the client application:

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

You create the following ACI in the `ou=people,dc=siroe,dc=com` entry:

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

Finally, you create an admin user on server three for server two to use for proxy authorization:

```
dn: cn=server3 proxy admin,cn=config
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

```
cn: server3 proxy admin
sn: server3 proxy admin
userPassword: secret
description: Entry for use by database links
```

You add the same proxy authorization ACI to server three as you did on server two. Add the following proxy authorization ACI to the `l=Zanzibar, ou=people,dc=siroe,dc=com` entry:

```
aci: (targetattr = "*")(version 3.0; acl "Proxied authorization for
 database links"; allow (proxy) userdn = "ldap:///cn=server3 proxy
 admin,cn=config";)
```

This ACI gives the server3 proxy admin read-only access to the data contained on the remote server within the `l=Zanzibar,ou=people,dc=siroe,dc=com` subtree only.

You then need to create an ACI somewhere above the `l=Zanzibar,ou=people,dc=siroe,dc=com` subtree that corresponds to the original client application. Use the same ACI as the one you created for the client on server two:

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

Using Referrals

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

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

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

Setting Default Referrals

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

Setting a Default Referral Using the Console

Set a default referral to your directory as follows:

1. On the Directory Server Console, select the Configuration tab.
2. Select the top entry in the navigation tree in the left pane.
3. Select the Settings tab in the right pane.
4. Enter an LDAP URL in the “Referrals to” text box and click OK.

   For example:

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

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

   ```
   "ldap://d1.siroe.com:389/dc=siroe,dc=com" "ldap://d2.siroe.com/"
   ```

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

Setting a Default Referral From the Command Line

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

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

```
cd /usr/iplanet/servers/shared/bin
```

Then, run the `ldapmodify` utility as follows:

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

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

Next, you add the default referral to the Zanzibar.com server:

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

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

Creating Smart Referrals

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

For example, a client application requests the following directory entry: `uid=bjensen,ou=people,dc=siroe,dc=com.` You return a smart referral to the client which points to the entry `cn=babs jensen,o=people,l=europe,dc=siroe,dc=com` on the server `directory.europe.siroe.com`.

The way the directory uses smart referrals conforms to the standard specified in RFC 2251 section 4.1.11. For more information, go to http://www.ietf.org/rfc/rfc2251.txt to read the RFC.

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

Creating Smart Referrals Using the Directory Server Console

1. On the Directory Server Console, select the Directory tab.
2. Browse through the tree in the left navigation pane and select the entry for which you want to add the referral.
3. Double-click the entry.

   Depending upon the entry you click, you will see either the Edit Entry dialog box or the Property Editor dialog box.

   If the Edit Entry dialog box is displayed, click the Advanced button. The Property Editor dialog box is displayed.
4. Click in one of the cells next to the “Object class” attribute and click Add Value.

   The Add Object Class dialog box displays.

5. Select `referral` from the list and click OK.
6. Click Add Attribute.

   The Add Attribute dialog box is displayed.
7. Scroll down the list of attributes to the `ref` attribute. Select the `ref` attribute, then click OK.

   The `ref` attribute now appears in the Property Editor dialog box.
8. In the text box next to the `ref` attribute, enter the LDAP URL to which you want to refer client application requests in the following format: `ldap://`*servername:portnumber*`/[`*optional_dn*`]` in the ref text box.

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

   ```
   ldap://directory.siroe.com:389/cn=joe,ou=people,dc=siroe,dc=com
   ```

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

   `ldap://`*servername:portnumber*`/`
9. Click OK to save your changes.

Creating Smart Referrals From the Command Line

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

To create a smart referral, create the relevant directory entry and add the `Referral` object class. This object class allows a single attribute, `ref`. The `ref` attribute is expected to contain an LDAP URL.

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

```
dn: uid=bjensen,ou=people,dc=siroe,dc=com
objectclass: referral
ref: ldap://directory.europe.siroe.com/cn=babs%20jensen,ou=people,
 l=europe,dc=siroe,dc=com
```

NOTE Any information after a space in an LDAP URL is ignored by the server. For this reason, you must use `%20` instead of spaces in any LDAP URL you intend to use as a referral.

To add the entry `uid=ssarette,ou=people,dc=siroe,dc=com` with a referral to `directory.europe.siroe.com`, you would include the following in your LDIF file before importing:

```
dn: uid=ssarette, ou=people, dc=siroe,dc=com
objectclass: top
objectclass: person
objectclass: organizationalperson
objectclass: inetOrgPerson
objectclass: referral
cn: somi sarette
sn: sarette
uid: ssarette
ref: ldap://directory.europe.siroe.com/cn=somi%20sarette,ou=people,
 l=europe,dc=siroe,dc=com
```

Use the `-M` option with `ldapmodify` when there is already a referral in the DN path. For information on the `-M` option, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

For more information on smart referrals, see *iPlanet Directory Server Deployment Guide*. For more information about the `ldapmodify` utility, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

Creating Suffix Referrals

The following procedure describes creating a referral in a suffix. This means that the suffix processes operations using a referral rather than a database or database link. For more information about referrals, refer to *iPlanet Directory Server Deployment Guide*.

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

Creating Suffix Referrals Using the Console

To create a suffix referral using the console:

1. On the Directory Server Console select the Configuration tab.
2. Under Data in the left pane, click the suffix to which you want to add a referral.

3. On the Suffix Settings tab, select one of the following radio buttons:

   Use Referrals. This means that a referral will be returned when this suffix receives any request from a client application.

   Use Referral on Update. This means a referral will be returned when this suffix receives an update request from a client application. This option is used to redirect update and write requests made by client applications to a read-only database.

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

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

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

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

6. Click Save.

Creating Suffix Referrals From the Command Line

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

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

```
cd /usr/iplanet/servers/shared/bin
```

Then, run `ldapmodify` as follows:

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

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

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

```
dn: cn="ou=people,dc=siroe,dc=com",cn=mapping tree,cn=config
objectclass: extensibleObject
objectclasss: nsmappingtree
nsslapd-state: referral
nsslapd-referral: ldap://zanzibar.com/
```

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

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

For more information about the suffix configuration attributes, refer to Table 3-1, “Suffix Attributes,” on page 76.

Chapter 4

Populating Directory Databases

Databases contain the directory data managed by your directory server. This chapter describes the following procedures for populating your directory databases:

- Importing Data
- Exporting Data
- Backing Up and Restoring Data
- Enabling and Disabling Read-Only Mode

Importing Data

iPlanet Directory Server provides three methods for importing data:

- Import from the Directory Server Console.

  You can use the Directory Server Console to append data to all of your databases, including database links.

- Initialize databases.

  You can use the Directory Server Console to import data to one database. This method overwrites any data contained by the database.

- Importing data from the command line.

  You can import data using the command-line utilities.

NOTE All LDIF files you import must use UTF-8 character set encoding.

The following table describes the differences between an import and initializing databases:

Table 4-1 Import Method Comparison

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

The following sections describe importing data:

- "Performing an Import From the Console," on page 134
- "Initializing a Database From the Console," on page 136
- "Importing From the Command Line," on page 137

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

Performing an Import From the Console

When you perform an import operation from the Directory Server Console, an `ldapmodify` operation is executed to append data, as well as to modify and delete entries. The operation is performed on all of the databases managed by your Directory Server, and on remote databases to which your Directory Server has a configured database link.

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

To import data from the Directory Server Console:

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

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

   The Import Database dialog box is displayed.

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

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

3. In the Options box, select one or more of the following options:

   Add Only. The LDIF file may contain modify and delete instructions in addition to the default add instructions. If you want the server to ignore operations other than add, select the “Add only” check box.

   Continue on Error. Select the “Continue on error” checkbox if you want the server to continue with the import even if errors occur. For example, you might use this option if you are importing an LDIF file that contains some entries that already exist in the database in addition to new ones. The server notes existing entries in the rejects file while adding all new entries.

4. In the “File for Rejects” field, enter the full path to the file in which you want the server to record all entries it cannot import or click Browse to select the file which will contain the rejects.

   For example, the server cannot import an entry that already exists in the database or an entry that has no parent object. The console will write the error message sent by the server to the rejects file.

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

5. Click OK.

   The server performs the import and also creates indexes.

Initializing a Database From the Console

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

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

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

To initialize a database using the Directory Server Console:

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

   You can also select Initialize Database from the Object menu.
4. In the “LDIF file” field, enter the full path to the LDIF file you want to import, or click Browse to locate it on your machine.
5. If you are operating the console from a machine local to the file being imported, skip to step 6. If you are operating the console from a machine remote to the server containing the LDIF file, select one of the following options:

   From local machine. Indicates that the LDIF file is located on the local machine.

   From server machine. Indicates that the LDIF file is located on a remote server. By default, the console looks for the file in the following directory: `/usr/iplanet/servers/slapd-`*ServerID*`/ldif`
6. Click OK.

Importing From the Command Line

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

- Using `ldif2db`.

  This import method overwrites the contents of your database and requires the server to be stopped.

- Using `ldif2db.pl`.

  This import method overwrites the contents of your database while the server is still running.

- Using `ldif2ldap`.

  This method appends your LDIF file through LDAP. You can use this method to append data to all of your databases.

Importing Using the ldif2db Command-Line Script

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

By default, the script first saves and then merges any existing `o=NetscapeRoot` configuration information with the `o=NetscapeRoot` configuration information in the files being imported.

CAUTION This script overwrites the data in your database.

To import LDIF with the server stopped:

1. From the command line, change to the following directory:
   `/usr/iplanet/servers/slapd-`*serverID*`/`

   where *serverID* is the name of your directory server.

2. Stop the server by typing the following:
   `./stop-slapd`

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

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

CAUTION If you a specify a database in the `-n` option that does not correspond with the suffix contained by the LDIF file, all of the data contained by the database is deleted and the import fails. Make sure that you do not misspell the database name.

Two examples of performing an import using `ldif2db` follow:

Windows NT batch file:

```
ldif2db.bat -n Database1
 -i c:\iplanet\servers\slapd-dirserver\ldif\demo.ldif
 -i c:\iplanet\servers\slapd-dirserver\ldif\demo2.ldif
```

UNIX shell script:

```
ldif2db -n Database1
 -i /usr/iplanet/servers/slapd-dirserver/ldif/demo.ldif
 -i /usr/iplanet/servers/slapd-dirserver/ldif/demo2.ldif
```

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

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

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

Importing Using the ldif2db.pl Perl Script

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

CAUTION This script overwrites the data in your database.

1. From the command line, change to the following directory:
   `/usr/iplanet/servers/slapd-`*`serverID`*`/`

   where *serverID* is the name of your directory server.

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

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

Two examples of performing an import using `ldif2db.pl` follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl ldif2db.pl -D "cn=Directory Manager"
-w secretpwd -i c:\iPlanet\servers\slapd-dirserver\ldif\demo.ldif
-n Database1
```

CAUTION You need to run the script from the following directory on NT machines: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

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

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

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

Importing Using the ldif2ldap Command-Line Script

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

To import LDIF using `ldif2ldap`:

1. From the command line, change to the following directory:
   `/usr/iplanet/servers/slapd-`*serverID/*

   where *serverID* is the name of your directory server.

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

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

Two examples of performing an import using `ldif2ldap` follow:

Windows NT batch file:

```
ldif2ldap "cn=Directory Manager" secret
c:\iplanet\servers\slapd-dirserver\ldif\demo.ldif
```

UNIX shell script:

```
ldif2ldap "cn=Directory Manager" secret
/usr/iplanet/servers/slapd-dirserver/ldif/demo.ldif
```

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

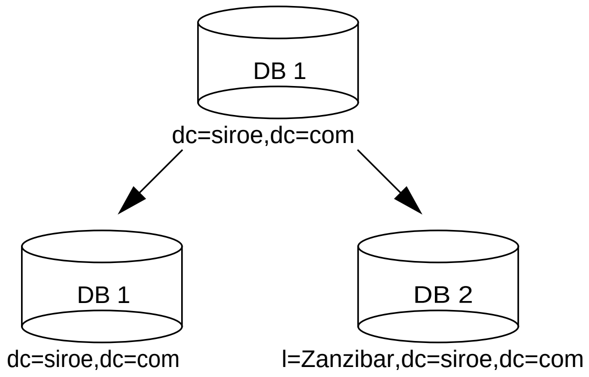
Exporting Data

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

Exporting data can be useful for the following:

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

For example, suppose your directory is contained by one database and you decide to split its contents over two databases as follows:



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

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

- "Exporting Directory Data to LDIF Using the Console," on page 142
- "Exporting a Single Database to LDIF Using the Console," on page 143
- "Exporting to LDIF From the Command Line," on page 143

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

CAUTION Do not stop the server during an export operation.

Exporting Directory Data to LDIF Using the Console

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

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

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

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

   The Export Database dialog box is displayed.

2. Enter the full path and file name of the LDIF file in the “LDIF File” field, or click Browse to locate the file.

   Browse is not enabled if you are running the console on a remote server. When the Browse button is not enabled, the file is stored by default in the `/usr/iplanet/servers/slapd-`*ServerID*`/ldif` directory.

3. If you are running the console on a machine remote to the server, two radio buttons are displayed beneath the LDIF file field. Select “To local machine” to indicate that you are exporting to an LDIF file in the machine from which you run the console. Select “To server machine” to indicate that you are exporting to an LDIF file located on the server’s machine.

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

   If you want to export only a single subtree of the suffix contained by the database, select the “Subtree” radio button and then enter the name of the suffix in the Subtree text box. This option allows you to export a subtree that is contained by more than one database.

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

5. Click OK to export the file.

Exporting a Single Database to LDIF Using the Console

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

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

   You can also select Export Database from the Object menu.

   The Export Partition dialog box is displayed.
4. In the "LDIF file" field, enter the full path to the LDIF file, or click Browse to locate it on your machine.

   When the Browse button is not enabled, by default the file is stored in the `/usr/iplanet/servers/slapd-`*ServerID*`/ldif` directory.
5. Click OK to export the file.

Exporting to LDIF From the Command Line

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

To export to LDIF from the command line:

1. From the command line, change to the following directory: `/usr/iplanet/servers/slapd-`*serverID*`/`

   where *serverID* is the name of your directory server.
2. Run the `db2ldif` command line script.

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

Windows NT batch file:

```
db2ldif -n database1 -a output.ldif -s "dc=siroe,dc=com" -s
 "o=NetscapeRoot"
```

UNIX shell script:

```
db2ldif -n database1 -a output.ldif -s "dc=siroe,dc=com" -s
 "o=NetscapeRoot"
```

The following table describes the `db2ldif` options used in the examples:

Option Name	Description
`-n`	Specifies the name of the database from which the file is being exported.
`-a`	Defines the output file in which the server saves the exported LDIF. This file is stored by default in the directory where the command-line script resides.
`-s`	Specifies the suffix or suffixes to include in the export. You may use multiple `-s` arguments.

Backing Up and Restoring Data

You can back up and restore your databases using the Directory Server Console or a command-line script.

The following sections describe the procedures for backing up and restoring data:

- "Backing Up All Databases," on page 145
- "Backing Up a Single Database," on page 146
- "Backing Up the dse.ldif Configuration File," on page 147
- "Restoring All Databases," on page 147
- "Restoring a Single Database," on page 150
- "Restoring Databases that Include Replicated Entries," on page 150
- "Restoring the dse.ldif Configuration File," on page 151

CAUTION Do not stop the server during a backup or restore operation.

Backing Up All Databases

The following procedures describe backing up all of the databases in your directory using the Directory Server Console and from the command line.

NOTE You cannot use these backup methods to back up the data contained by databases on a remote server that you chain to using database links.

Backing Up All Databases From the Server Console

When you back up your databases from the Directory Server Console, the server copies all of the database contents and associated index files to a backup location. You can perform a backup while the server is running.

To back up your databases from the Server Console:

1. On the Directory Server Console select the Tasks tab.
2. Click "Back Up Directory Server."

   The Backup Directory dialog box is displayed.
3. Enter the full path of the directory where you want to store the backup file in the Directory text box, or click "Use default" and the server provides a name for the backup directory.

   If you are running the console on the same machine as the directory, you can also click Browse to locate a local directory.

   If you choose to use the default, the backup files will be placed in the following location:

   `/usr/iplanet/servers/slapd-`*`serverID`*`/bak/`*backup_directory*

   where *`serverID`* is the name of your directory server. The *backup_directory* variable names a directory using the name of the backup file. By default, the backup directory name contains the time and date the backup was created (`YYYY_MM_DD_hhmmss`).
4. Click OK to create the backup.

Backing Up All Databases From the Command Line

You can back up your databases from the command line using the `db2bak` command-line script. This script works when the server is running or when the server is stopped.

You cannot back up the configuration information using this backup method. For information on backing up the configuration information, refer to “Backing Up the dse.ldif Configuration File,” on page 147.

To back up your directory from the command line using the `db2bak` script:

1. At the command prompt, change to the following directory:
   `/usr/iplanet/servers/slapd-`*serverID*

   where *serverID* is the name of your directory server.

2. Run the `db2bak` command-line script.

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

Two examples of performing an import using `db2bak` follow:

Windows NT batch file:

```
db2bak \usr\iplanet\servers\slapd-dirserver\bak\bak_20010701103056
```

UNIX shell script:

```
db2bak /usr/iplanet/servers/slapd-dirserver/bak/bak_20010701103056
```

You can specify the backup directory and output file where the server saves the exported LDIF file. If you do not specify a directory and output file, the directory will store the file by default in the directory where the command-line script resides. By default, the backup file is named according to the year-month-day-hour format (`YYYY_MM_DD_hhmmss`).

Backing Up a Single Database

To back up a single database:

1. At the command prompt, change to: `/usr/iplanet/servers/slapd-`*serverID*

   where *serverID* is the name of your directory server.

2. If the server is running, type the following to stop it:
   `./stop-slapd`

3. Change to the directory containing the database you want to back up:
   `cd db/`*database_name*

4. Copy all of the files in the directory to a backup directory you create. Do not use the `/usr/iplanet/servers/slapd-`*serverID*`/bak/` directory, as the Directory Server Console assumes the backups contained by this directory are global.

NOTE You cannot use this backup method to back up the data contained by databases on remote servers (the databases chained to by the database links).

Backing Up the dse.ldif Configuration File

Directory Server automatically backs up the `dse.ldif` configuration file. When you start your directory server, the directory creates a backup of the `dse.ldif` file automatically in a file named `dse.ldif.startOK` in the `/usr/iplanet/servers/slapd-`*serverID*`/config` directory.

When you make modifications to the `dse.ldif` file, the file is first backed up to a file called `dse.ldif.bak` in the `/usr/iplanet/servers/slapd-`*serverID*`/config` directory before the directory writes the modifications to the `dse.ldif` file.

Restoring All Databases

The following procedures describe restoring all of the databases in your directory using the Directory Server Console and from the command line.

NOTE While restoring databases, the server must be running. However, the databases will be unavailable for processing operations during the restore.

Restoring All Databases from the Console

If your databases become corrupted, you can restore data from a previously generated backup using the Directory Server Console. This process consists of stopping the server and then copying the databases and associated index files from the backup location to the database directory.

CAUTION Restoring your databases overwrites any existing database files.

To restore your databases from a previously created backup:

1. On 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, `/usr/iplanet/servers/slapd-`*`serverID`*`/bak/`*backup_name*

   where *serverID* is the name of your directory server and *backup_name* is the name of the backup file.
4. Click OK to restore your databases.

Restoring Your Database From the Command Line

You can restore your databases from the command line by using the following scripts:

- Using the `bak2db` command-line script. This script requires the server to be shut down.
- Using the `bak2db.pl` perl script. This script works while the server is running.

Using the bak2db Command-Line Script

To restore your directory from the command line while the server is shut down:

1. At the command prompt, change to the following directory: `/usr/iplanet/servers/slapd-`*`serverID`*

   where *serverID* is the name of your directory server.
2. If the server is running, type the following to stop it:

   ```
   ./stop-slapd
   ```
3. Run the `bak2db` command-line script.

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

Two examples of performing an import using `bak2db` follow:

Windows NT batch file:

```
bak2db.bat
 \usr\iPlanet\servers\slapd-dirserver\bak\bak_20010701103056
```

UNIX shell script:

```
bak2db /usr/iplanet/servers/slapd-dirserver/bak/bak_20010701103056
```

The `bak2db` script requires that you define the full path and name of the input file.

Using bak2db.pl Perl Script

To restore your directory from the command line while the server is running:

1. At the command prompt, change to the following directory: `/usr/iplanet/servers/slapd-`*serverID*

   where *serverID* is the name of your directory server.

2. Run the `bak2db.pl` perl script.

   For more information on using this perl script, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Two examples of performing an import using `bak2db.pl` follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl bak2db.pl -D "cn=Directory Manager"
 -w secret
 -a \usr\iplanet\servers\slapd-dirserver\bak\mybak_20010701103056
```

CAUTION You need to run perl scripts from the following directory on Windows NT: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

```
bak2db.pl -D "cn=Directory Manager" -w secret
 -a /usr/iplanet/servers/slapd-dirserver/bak/mybak_20010701103056
```

The following table describes the `bak2db.pl` options used in the examples:

Option Name	**Description**
-a	Defines the full path and name of the input file.
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.

Restoring a Single Database

To restore a single database:

1. At the command prompt, change to the following directory:

   `cd /usr/iplanet/servers/slapd-`*serverID*

   where *serverID* is the name of your directory server.

2. If the server is running, type the following to shut it down:

   ```
   ./stop-slapd
   ```

3. Change to the directory containing the backup you want to restore.

4. Copy all of the files to the directory containing the database you want to overwrite with your backup. For example, you might type the following:

   `cp` *backup_file* `../db/`*database_name*

Restoring Databases that Include Replicated Entries

If you are restoring a database that is supplying entries to other servers, then you must reinitialize all of the servers that receive updates from the restored database (for example, consumer servers, hub servers, and, in multi-master replication environments, other supplier servers). The change log associated with the restored database will be erased during the restore operation. A message will be logged to the supplier servers' log files indicating that reinitialization is required.If you are restoring a database containing data received from a supplier server, then one of two situations can occur:

- Change log entries have not yet expired on the supplier server.

  If the supplier server change log has not expired since the database backup was taken, then you can restore the local consumer and continue with normal operations. This situation occurs only if the backup was taken within a period of time that is shorter than the value you have set for the maximum change log age attribute. This attribute is called `nsslapd-changelogmaxage` and can be found in the `cn=changelog5,cn=config` entry. For more information about this option, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

iPlanet Directory Server automatically detects the compatibility between the replica and its change log. If a mismatch is detected, the server removes the old change log file and creates a new, empty one.

- Change log entries have expired on the supplier server since the time of the local backup.

  If change log entries have expired, you need to initiate consumer reinitialization. For more information on reinitializing consumers, refer to "Initializing Consumers," on page 297.

For information on managing replication, see "Managing Replication," on page 269.

Restoring the dse.ldif Configuration File

To restore the `dse.ldif` configuration file, stop the server, then use the procedure outlined in "Restoring a Single Database," on page 150 to copy the backup copy of the `dse.ldif` file into your directory. Once you have copied the data, restart your server.

The directory creates two backup copies of the dse.ldif file in the `/usr/iplanet/servers/slapd-`*serverID*`/config` 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. Copy the file with the most recent changes to your directory.

Enabling and Disabling Read-Only Mode

Before performing certain operations of export or backup on your Directory Server, you can enable read-only mode on any of the databases to ensure you have a faithful image of the state of these databases at a given time.

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

Enabling Read-Only Mode

1. On the Directory Server Console, select the Configuration tab, and expand the Data folder in the navigation tree.
2. Select the database that you want to place in read-only mode, and click the Database Settings tab in the right pane.
3. Select the “Database is Read-Only” checkbox.
4. Click Save.

   Your change takes effect immediately.

Before performing an import or restore operation, you should ensure that the databases affected by the operation are *not* in read-only mode. If they are, use the following procedure to make them available for updates.

Disabling Read-Only Mode

1. On the Directory Server Console, select the Configuration tab, and expand the Data tree.
2. Select the database that you want to make available for updates, and click the Database Settings tab in the right pane.
3. Clear the “Database is Read-only” checkbox.
4. Click Save.

   Your change takes effect immediately.

Chapter 5

Advanced Entry Management

You can group the entries contained by your directory to simplify the management of user accounts. Directory Server supports a variety of methods for grouping entries and sharing attributes between entries.

This chapter describes the following grouping mechanisms and their procedures:

- Using Groups
- Using Roles
- Assigning Class of Service

To take full advantage of the features offered by roles and class of service, in the planning phase of your directory deployment determine your directory topology. Refer to the *iPlanet Directory Server Deployment Guide* for more information.

Using Groups

Groups are a mechanism for associating entries for ease of administration. This mechanism was provided with previous versions of Directory Server and should be used primarily for compatibility with older versions of the server.

The following sections describe managing static and dynamic groups. For a conceptual overview of groups, refer to the *iPlanet Directory Server Deployment Guide*. For more information about administering groups, refer to *Managing Servers with Directory Console*.

Managing Static Groups

Static groups allow you to group entries by specifying the same group value in the DN attribute of any number of users. This section includes the following procedures for creating and modifying static groups:

- "Adding a New Static Group," on page 154
- "Modifying a Static Group," on page 155

NOTE	If you have a user that is remote from the definition of the static group, then you can use the referential integrity plug-in to ensure that deleted user entries are automatically deleted from the static group. For more information about using referential integrity with chaining, refer to "Configuring the Chaining Policy," on page 88.

Adding a New Static Group

1. In the Directory Server Console, select the Directory tab.
2. Right-click the entry in the left pane under which you want to add a new group. Select New > Group.

   You can also go to the Object menu and select New > Group.
3. Click General in the left pane. Type a name for your new group in the "Group Name" field.

   The group name is required.
4. Enter a description of the new group in the "Description" field.
5. Click Members in the left pane. In the right pane, select the Static Group tab. Click Add to add new members to the group.

   The standard "Search users and groups" dialog box appears.
6. In the Search drop-down list, select what sort of entries to search for (users, groups, or both) then click Search. Select one of the entries returned and click OK.
7. Click Languages in the left pane to add language-specific information for your group.
8. Click OK to create your new group. It appears in the right pane.

Modifying a Static Group

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

   The directory contents appear in the left pane.

2. Double-click the entry you want to modify or select Open from the Object menu.

   The Edit Group dialog box appears.

3. Make your changes to the group information. Click OK.

   To view your changes, go to the View menu and select Refresh.

Managing Dynamic Groups

Dynamic groups filter users based on their DN and include them in a single group. This section contains the following procedures for creating and modifying dynamic groups:

- "Adding a New Dynamic Group," on page 155
- "Modifying a Dynamic Group," on page 155

Adding a New Dynamic Group

1. Follow steps 1-4 of "Adding a New Static Group," on page 154.

2. Click Members in the left pane. In the right pane, select the Dynamic Group tab. Click Add to create a LDAP URL for querying the database.

   The standard "Construct and Test LDAP URL" dialog box displays.

3. Enter an LDAP URL in the text field or select Construct to be guided through the construction of an LDAP URL.

4. Click Languages in the left pane to add language-specific information for your group.

5. Click OK to create your new group.

   Your new group appears in the right pane.

Modifying a Dynamic Group

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

   The directory contents appear in the left pane.

2. Double-click the entry you want to modify or select Open from the Object menu.

   The Edit Group dialog box appears.

3. Make your changes to the group information. Click OK.

   To view your changes, go to the View menu and select Refresh.

Using Roles

Roles are a new entry grouping mechanism that unify the static and dynamic groups described in the previous sections. Roles are designed to be more efficient and easier to use for applications. For example, an application can locate the role of an entry, rather than select a group and browse the members list.

This section contains the following topics:

- "About Roles," on page 156
- "Managing Roles Using the Console," on page 157
- "Managing Roles Using the Command Line," on page 162
- "Using Roles Securely," on page 165

About Roles

Roles unify the static and dynamic group concept supported by previous versions of Directory Server.

You can use roles to:

- Enumerate the members of a role.

  Having an enumerated list of role members can be useful for resolving queries for role members quickly.

- Determine whether a given entry possesses a particular role.

  Knowing the roles possessed by an entry can help you determine whether the entry possesses the target role.

- Enumerate all the roles possessed by a given entry.
- Assign a particular role to a given entry.

- Remove a particular role from a given entry.

You can do everything you would normally do with static groups with managed roles, and you can filter members using filtered roles as you used to do with dynamic groups. Roles are easier to use than groups, more flexible in their implementation, and reduce client complexity.

However, evaluating roles is more resource intensive because the server does the work for the client application. With roles, the client application can check role membership by searching the `nsRole` attribute. The `nsRole` attribute is a computed attributed that is not stored with the entry itself, which identifies which roles an entry belongs to. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

Each role has *members*, or entries that possess the role. You can specify members either explicitly or dynamically. How you specify role membership depends upon the type of role you are using. Directory Server supports three types of roles:

- Managed roles.

  A managed role allows you to create an explicit enumerated list of members.

- Filtered roles.

  A filtered role allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

- Nested roles.

  A nested role allows you to create roles that contain other roles.

For more information about how roles work, refer to *iPlanet Directory Server Deployment Guide*.

Managing Roles Using the Console

This section contains the following procedures for creating and modifying roles:

- "Creating a Managed Role," on page 158
- "Creating a Filtered Role," on page 159
- "Creating a Nested Role," on page 160
- "Viewing and Editing an Entry's Roles," on page 160
- "Modifying a Role Entry," on page 161

- "Making a Role Inactive," on page 161
- "Reactivating a Role," on page 162
- "Deleting a Role," on page 162

When you create a role, you need to decide whether a user can add themselves or remove themselves from the role. Refer to "Using Roles Securely," on page 165 for more information about roles and access control.

Creating a Managed Role

Managed roles allow you to create an explicit enumerated list of members. Managed roles are added to entries by adding the `nsRoleDN` attribute to the entry.

To create and add members to a managed role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry for your new role.
3. Go to the Object menu and select New > Role.

   You can also right click the entry and select New > Role.

   The Create New Role dialog box is displayed.
4. Click General in the left pane. Type a name for your new role in the "Role Name" field.

   The role name is required.
5. Enter a description of the new role in the "Description" field.
6. Click Members in the left pane.

   A search dialog box appears briefly.
7. In the right pane, select Managed Role. Click Add to add new entries to the list of members.

   The standard "Search users and groups" dialog box appears.
8. In the Search drop-down list, select Users from the Search drop-down list, then click Search. Select one of the entries returned and click OK.
9. When you have finished adding entries to the role, click OK.

   The new role appears in the right pane.

Creating a Filtered Role

You assign entries to a filtered role depending upon a particular attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

To create and add members to a filtered role:

1. Follow steps 1-5 of "Creating a Managed Role," on page 158.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Filtered Role.
4. Enter an LDAP filter in the text field, or click Construct to be guided through the construction of an LDAP filter.
5. If you click Construct, the standard LDAP URL construction dialog appears. Disregard the fields identifying the host server, port number and base DN.

   c. From the "Search" drop-down list, select whether the filter searches for entries one level below the base DN, or searches the whole subtree beneath the base DN. Do not use the base DN only option, because roles are designed to work only on entries within their scope.

   d. Select the types of entries you want to filter from the "For" drop-down list.

      You can choose between users, groups, or both.

   e. Select an attribute from the "Where" drop-down list. The two fields following it allow you to refine your search by selecting one of the qualifiers from the drop-down list (such as contains, does not contain, is, is not) and enter an attribute value in the text box. To add additional filters, click More. To remove unnecessary filters, click Fewer.

   f. Click OK to save your filter.
6. Click Test to try your filter.

   A Filter Test Result dialog box displays the entries matching your filter.
7. Click OK.

   The new role appears in the right pane.

Creating a Nested Role

Nested roles allow you to create roles that contain other roles. Before you create a nested role, another role must exist. When you create a nested role, the console displays a list of the roles available for nesting. The roles nested within the nested role are specified using the `nsRoleDN` attribute.

To create and add members to a nested role:

1. Follow steps 1-5 of "Creating a Managed Role," on page 158.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Nested Role.
4. Click Add to add roles to the list.The members of the nested role are members of other existing roles.

   The Role Selector dialog box appears.
5. Select a role from the "Available roles" list and click OK.
6. Click OK.

   The new role appears in the right pane.

Viewing and Editing an Entry's Roles

To view or edit a role associated with an entry from the console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the entry for which you want to view or edit a role. Select Set Roles from the Object menu.

   The Roles dialog box displays.
3. Select the Managed Roles tab to display the managed roles to which this entry belongs.
4. To add a new managed role, click Add and select an available role from the Role Selector window. Click OK.

   To remove a managed role, select it and click Remove.

   To edit a managed role associated with an entry, click Edit. The Edit Entry dialog box displays. Make any changes to the general information or members and click OK.

5. Select the Other Roles tab to view what filtered or nested roles this entry belongs to.
6. Click Edit to make changes to any filtered or nested roles associated with the entry. Click OK to save your changes.
7. Click OK once you have finished modifying the roles to save your changes.

Modifying a Role Entry

To edit an existing role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Double-click the role.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the role name and description.
5. Click Members in the left pane to change the members of managed and nested roles or to change the filter of a filtered role.
6. Click OK to save your changes.

Making a Role Inactive

You can temporarily disable the members of a role by inactivating the role to which they belong. Inactivating a role inactivates the entries possessed by the role and not the role itself.

To temporarily disable the members of a role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Select the role. Select Inactivate from the Object menu.

   You can also right-click the role and select Inactivate from the menu.

   The role is inactivated.

   To see the inactivated entries, select Inactivation State from the View menu. A red slash through the role icon indicates that the role has been inactivated.

Reactivating a Role

To reactivate a disabled role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Select the role. Select Activate from the Object menu.

   You can also right-click the role and select Activate from the menu.

   The role is reactivated.

   To see inactivated entries, select Inactivation State from the View menu.The role icon appears as normal, indicating that the role is active.

Deleting a Role

Deleting a role deletes the role only, not its members.

To delete a role:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Right-click the role and select Delete.

   A dialog box appears asking you to confirm the deletion. Click Yes.
4. The Deleted Entries dialog box appears to inform you that the role was successfully deleted. Click OK.

Managing Roles Using the Command Line

Roles inherit from the `ldapsubentry` object class, which is defined in the ISO/IEC X.509 standard. In addition, each type of role has two specific object classes that inherits from the `nsRoleDefinition` object class. Once you create a role, you assign members to it as follows:

- Members of a managed role have the `nsRoleDN` attribute in their entry.
- Members of a filtered role are entries that match the filter specified in the `nsRoleFilter` attribute.

- Members of a nested role are members of the roles specified in the `nsRoleDN` attributes of the nested role definition entry.

The following table lists the new object classes and attributes associated with each type of role:

Role Type	Object Classes	Attributes
Managed Role	nsSimpleRoleDefinition nsManagedRoleDefinition	Description (optional)
Filtered Role	nsComplexRoleDefinition nsFilteredRoleDefinition	nsRoleFilter Description (optional)
Nested Role	nsComplexRoleDefinition nsNestedRoleDefinition	nsRoleDN Description (optional)

NOTE In some cases you need to protect the value of the `nsRoleDN` attribute with an ACI, as the attribute is writable. For more information about security and roles, refer to "Using Roles Securely," on page 165.

Examples: Managed Role Definition

You want to create a role to be assigned to all marketing staff. Run the `ldapmodify` script as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

Specify the managed role as follows:

```
dn: cn=Marketing,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: Marketing
description: managed role for marketing staff
```

Notice that the `nsManagedRoleDefinition` object class inherits from the `LDAPsubentry`, `nsRoleDefinition` and `nsSimpleRoleDefinition` object classes.

Assign the role to a marketing staff member named Bob by doing an `ldapmodify` as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
dn: cn=Bob,ou=people,dc=siroe,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=Marketing,ou=people,dc=siroe,dc=com
```

The `nsRoleDN` attribute present in the entry indicates that the entry is a member of a managed role, the marketing managed role `cn=Marketing,ou=people,dc=siroe,dc=com`.

Example: Filtered Role Definition

You want to set up a filtered role for sales managers. Run the `ldapmodify` script as follows:

```
ldapmodify -D "cn=Directory Manager" -w secret -h host -p 389
```

Specify the filtered role as follows:

```
dn: cn=SalesManagerFilter,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: SalesManagerFilter
nsRoleFilter: o=sales managers
Description: filtered role for sales managers
```

Notice that the `nsFilteredRoleDefinition` object class inherits from the `LDAPsubentry`, `nsRoleDefinition`, and `nsComplexRoleDefinition` object classes. The `nsRoleFilter` attribute specifies the `o` (organization) attributes that contain the value of `sales managers`.

The following entry matches the filter (possesses the `o` attribute with the value `sales manager`) and therefore is a member of this filtered role:

```
dn: cn=Pat,ou=people,dc=siroe,dc=com
objectclass: person
cn: Pat
sn: Pat
userPassword: bigsecret
o: sales managers
```

Example: Nested Role Definition

You want to create a role that contains both the marketing staff and sales managers contained by the roles you created in the previous examples. The nested role you create using `ldapmodify` appears as follows:

```
dn: cn=MarketingSales,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsNestedRoleDefinition
cn: MarketingSales
nsRoleDN: cn=SalesManagerFilter,ou=people,dc=siroe,dc=com
nsRoleDN: cn=Marketing,ou=people,dc=siroe,dc=com
```

Notice the `nsNestedRoleDefinition` object class inherits from the `LDAPsubentry`, `nsRoleDefinition`, and `nsComplexRoleDefinition` object classes. The `nsRoleDN` attributes contain the DN of the marketing managed role and the sales managers filtered role.

Both of the users in the previous examples, Bob and Pat, would be members of this new nested role.

Using Roles Securely

Not every role is suitable for use in a security context. When creating a new role, consider how easily the role can be assigned to and removed from an entry. Sometimes it is appropriate for users to be able to easily add themselves to or remove themselves from a role. For example, if you had an interest group role called Mountain Biking, you would want interested users to add themselves or remove themselves easily.

However, in some security contexts it is inappropriate to have such open roles. For example, consider account inactivation roles. By default, account inactivation roles contain ACIs defined for their suffix. When creating a role, the server administrator decides whether a user can assign themselves to or remove themselves from the role.

For example, user A possesses the managed role, MR. The MR role has been locked using account inactivation through the command line. This means that user A cannot bind to the server because the `nsAccountLock` attribute is computed as "true" for that user. However, suppose the user was already bound and noticed that he is now locked through the MR role. If there are no ACIs preventing him, the user can remove the `nsRoleDN` attribute from his entry and unlock himself.

To prevent users from removing the `nsRoleDN` attribute, use the following ACIs depending upon the type of role being used.

Managed roles. For entries that are members of a managed role, use the following ACI to prevent users from unlocking themselves by removing the appropriate `nsRoleDN`:

```
aci: (targetattr=”nsRoleDN”)
     (targattrfilters=”

  add=nsRoleDN:(!(nsRoleDN=cn=AdministratorRole,dc=siroe,dc=com)),

  del=nsRoleDN:(!(nsRoleDN=cn=nsManagedDisabledRole,dc=siroe,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, and modify the attribute used by the filtered role. If the value of the filter attribute is computed, then all attributes that can modify the value of the filter attribute should be protected in the same way.

Nested roles. A nested role is comprised of filtered and managed roles, so the above points should be considered for each of the roles that comprise the nested role.

For more information about account inactivation, see “Inactivating Users and Roles,” on page 263.

Assigning Class of Service

A class of service (CoS) allows you to share attributes between entries in a way that is transparent to applications. CoS simplifies entry management and reduces storage requirements.

There are two methods for creating and managing CoS, using the Directory Server console or through the command line. The following sections describe CoS in more detail and provide the procedures for managing CoS through both the console and the command line:

- “About CoS,” on page 167
- “Managing CoS Using the Console,” on page 171

- "Managing CoS From the Command Line," on page 174
- "Creating Role-Based Attributes," on page 181
- "Access Control and CoS," on page 182

About CoS

Clients of the Directory Server read the attributes on a user's entry. With CoS, some attribute values may not be stored with the entry itself. Instead, they are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of the following two types of entry in your directory:

- CoS Definition Entry.

  The CoS definition entry identifies the type of CoS you are using. Like the role definition entry, it inherits from the `LDAPsubentry` object class. The CoS definition entry is below the branch at which it is effective.

- Template Entry.

  The CoS template entry contains a list of the shared attribute values. Changes to the template entry attribute values are automatically applied to all the entries within the scope of the CoS. A single CoS might have more than one template entry associated with it.

The CoS definition entry and template entry interact to provide attribute information to their *target entries*, any entry within the scope of the CoS.

NOTE LDAP search requests containing a filter that references an attribute defined by a CoS may return unexpected results. Take care when deciding which attributes to generate using a CoS.

The following sections describe the entries that make up a CoS in more detail and provide examples of each type of CoS.

About the CoS Definition Entry

The CoS definition entry is an instance of the `cosSuperDefinition` object class. The CoS definition entry also contains an object class that specifies the type of template entry it uses to generate the entry. You can specify three different object classes depending upon the type of CoS you want to use. The target entries share the same parent as the CoS definition entry.

There are 3 types of CoS, defined using three types of CoS definition entries:

- Pointer CoS.

  A pointer CoS identifies the template entry using the template DN only.

- Indirect CoS.

  An indirect CoS identifies the template entry using the value of one of the target entry's attributes. For example, an indirect CoS might specify the `manager` attribute of a target entry. The value of the `manager` attribute is then used to identify the template entry.

  The target entry's attribute must be single-valued and contain a DN.

- Classic CoS.

  A classic CoS identifies the template entry using a combination of the template entry's base DN and the value of one of the target entry's attributes.

For more information about the object classes and attributes associated with each type of CoS, refer to "Managing CoS From the Command Line," on page 174.

If the CoS logic detects that an entry contains an attribute for which the CoS is generating values, by default the CoS supplies the client application with the attribute value in the entry itself. However, you can use the CoS definition entry to control this behavior.

About the CoS Template Entry

The CoS template entry contains the value or values of the attributes generated by the CoS logic. The CoS template entry contains a general object class of `cosTemplate`. The CoS template entries for a given CoS are stored in the directory tree along with the CoS definition.

The relative distinguished name (RDN) of the template entry is determined by one of the following:

- The DN of the template entry alone.

  This type of template is associated with a pointer CoS.

- The value of one of the target entry's attributes.

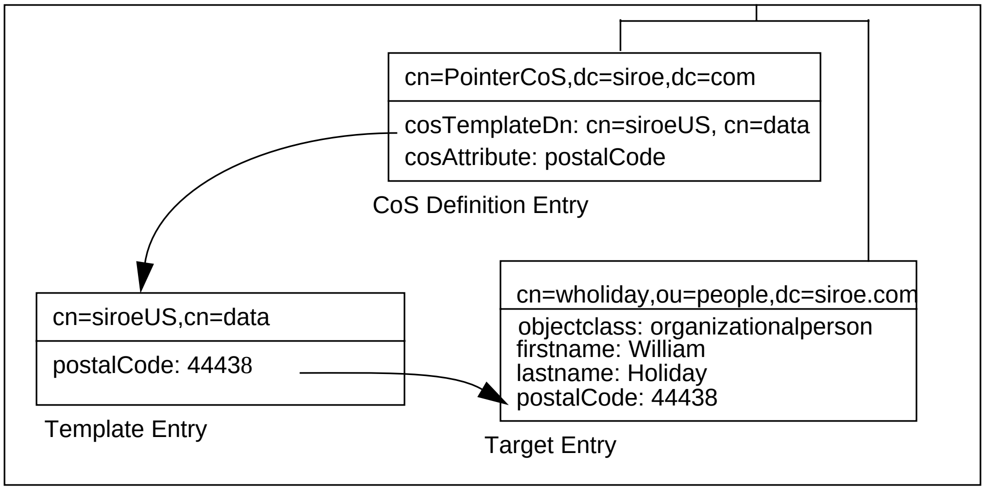
  The attribute used to provide the relative DN to the template entry is specified in the CoS definition entry using the `cosIndirectSpecifier` attribute. This type of template is associated with an indirect CoS.

- By a combination of the DN of the subtree where the CoS performs a one level search for templates and the value of one of the target entry's attributes.

This type of template is associated with a classic CoS.

How a Pointer CoS Works

You create a CoS that shares a common postal code with all of the entries stored under `dc=siroe,dc=com`. The three entries for this CoS appear as follows:

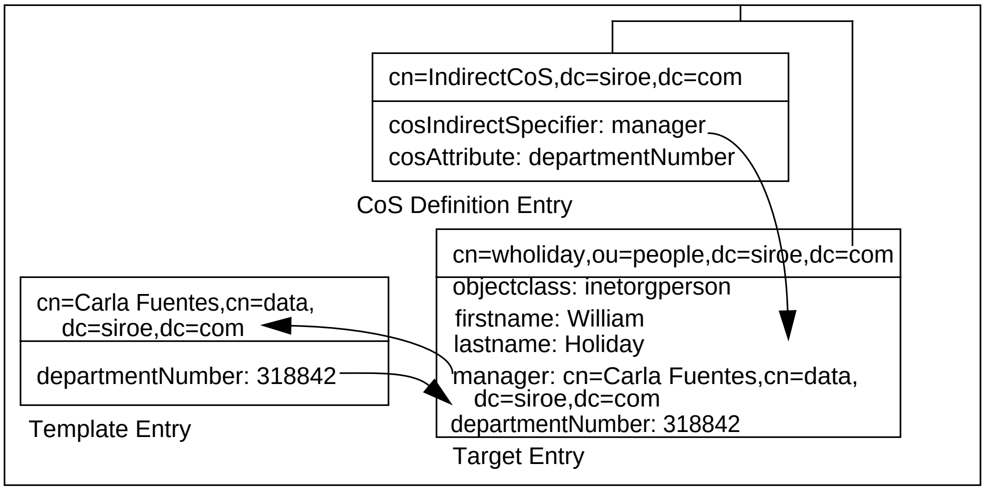


In this example, the template entry is identified by its DN, `cn=siroeUS,cn=data`, in the CoS definition entry. Each time the `postalCode` attribute is queried on the entry `cn=wholiday,ou=people,dc=siroe,dc=com`, the Directory Server returns the value available in the template entry `cn=siroeUS,cn=data`.

How an Indirect CoS Works

You can create an indirect CoS that uses the `manager` attribute of the target entry to identify the template entry.

The three CoS entries appear as follows:

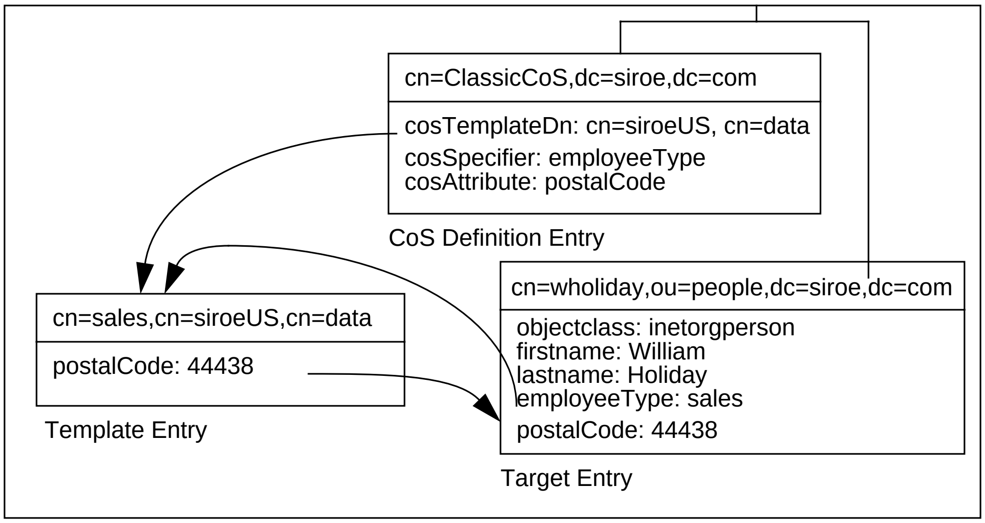

cn=IndirectCoS,dc=siroe,dc=com
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
CoS Definition Entry
cn=wholiday,ou=people,dc=siroe,dc=com
objectclass: inetorgperson
firstname: William
lastname: Holiday
manager: cn=Carla Fuentes,cn=data, dc=siroe,dc=com
departmentNumber: 318842
Target Entry
cn=Carla Fuentes,cn=data, dc=siroe,dc=com
departmentNumber: 318842
Template Entry

In this example, the target entry for William Holiday contains the indirect specifier, the `manager` attribute. William's manager is Carla Fuentes, so the `manager` attribute contains a pointer to the DN of the template entry, `cn=Carla Fuentes,ou=people,dc=siroe,dc=com`. The template entry in turn provides the `departmentNumber` attribute value of 318842.

How a Classic CoS Works

You can create a classic CoS that uses a combination of the template DN and a CoS specifier to identify the template entry containing the postal code.

The three CoS entries appear as follows:


cn=ClassicCoS,dc=siroe,dc=com
cosTemplateDn: cn=siroeUS, cn=data
cosSpecifier: employeeType
cosAttribute: postalCode
CoS Definition Entry
cn=sales,cn=siroeUS,cn=data
postalCode: 44438
Template Entry
cn=wholiday,ou=people,dc=siroe,dc=com
objectclass: inetorgperson
firstname: William
lastname: Holiday
employeeType: sales
postalCode: 44438
Target Entry

In this example, the Cos definition entry's `cosSpecifier` attribute specifies the `employeeType` attribute. This attribute, in combination with the template DN, identify the template entry as `cn=sales,cn=siroeUS,cn=data`. The template entry then provides the value of the `postalCode` attribute to the target entry.

Managing CoS Using the Console

This section describes creating and editing CoS through the Directory Server Console. It includes the following sections:

- "Creating a New CoS," on page 171
- "Editing an Existing CoS," on page 173
- "Deleting a CoS," on page 173

Creating a New CoS

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry for your new class of service.
3. Go to the Object menu and select New > Class of Service.

   You can also right click the entry and select New > Class of Service.

   The Create New Class of Service dialog displays.

4. Select General in the left pane. In the right pane, enter the name of your new class of service in the “Class Name” field. Enter a description of the class in the “Description” field.

5. Click Attributes in the left pane. The right pane displays a list of attributes generated on the target entries.

   Click Add to browse the list of possible attributes and add them to the list.

6. Once you have added an attribute to the list, a drop-down list appears in the “Class of Service Behavior” column.

   Select “Does not override target entry attribute” to tell the directory to only return a generated value if there is no corresponding attribute value stored with the entry.

   Select “Overrides target entry attribute” to make the value of the attribute generated by the CoS override the local value.

   Select “Overrides target entry attribute and is operational” to make the attribute override the local value and to make the attribute operational, so that it is not visible to client applications unless explicitly requested.

NOTE You can only make an attribute operational if it is also defined as operational in the schema.

For example, if your CoS generates a value for the `description` attribute, you cannot select “Overrides target entry attribute and is operational” because this attribute is not marked operational in the schema.

7. Click Template in the left pane. In the right pane, select how the template entry is identified.

   By its DN. If you choose to have the template entry identified by only its DN (a pointer CoS), enter the DN of the template in the “Template DN” field. Click Browse to locate the DN on your local server.

   Using the value of one of the target entry’s attribute. If you choose to have the template entry identified by the value of one of the target entry’s attributes (an indirect CoS), enter the attribute name in the “Attribute Name” field. Be sure to select an attribute which contains DN values. Click Change to select a different attribute from the list of available attributes.

Using both its DN and the value of one of the target entry's attributes. If you choose to have the template entry identified by both its DN and the value of one of the target entry's attributes (a classic CoS), enter both a template DN and an attribute name.

8. Click OK.

Editing an Existing CoS

The following procedure describes changing the description and attributes generated on the target entry of an existing class of service.

To edit an existing CoS:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Double-click the CoS.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the CoS name and description.
5. Click Attributes in the left pane to add or remove attributes generated by the CoS.
6. Click OK to save your changes.

   The target entries of the CoS are automatically updated.

Deleting a CoS

The following procedure describes deleting a CoS:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Right-click the CoS and select Delete. A dialog box appears asking you to confirm the deletion. Click Yes.
4. The Deleted Entries dialog box appears to inform you that the CoS was successfully deleted. Click OK.

Managing CoS From the Command Line

Because all configuration information and template data is stored as entries in the directory, you can use standard LDAP tools for CoS configuration and management. This section contains the following topics:

- "Creating the CoS Definition Entry From the Command Line," on page 174
- "Creating the CoS Template Entry From the Command Line," on page 177
- "Example of a Pointer CoS," on page 178
- "Example of an Indirect CoS," on page 179
- "Example of a Classic CoS," on page 180

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. The following table lists the object classes associated with each type of CoS definition entry:

Table 5-1 CoS Definition Entry Object Classes

CoS Type	Object Classes	Description
`Pointer CoS`	`cosPointerDefinition`	Identifies the template entry associated with the CoS definition using the template entry's DN value. The DN of the template entry is specified in the `cosTemplateDn` attribute.
`Indirect CoS`	`cosIndirectDefinition`	Identifies the template entry using the value of one of the target entry's attributes. The attribute of the target entry is specified in the `cosIndirectSpecifier` attribute.
`Classic CoS`	`cosClassicDefinition`	Identifies the template entry using both the template entry's DN (as specified in the `cosTemplateDn` attribute) and the value of one of the target entry's attributes (as specified in the `cosSpecifier` attribute).

You can use the following attributes in your CoS definition entries:

Table 5-2 CoS Definition Entry Attributes

Attribute	Definition
`cosAttribute`	Provides the name of the attribute for which you want to generate a value. You can specify more than one `cosAttribute` value. This attribute is used by all types of CoS definition entries.
`cosIndirectSpecifier`	Specifies the attribute value used by an indirect CoS to identify the template entry.
`cosSpecifier`	Specifies the attribute value used by a classic CoS, which, along with the template entry's DN, identifies the template entry.
`cosTemplateDn`	Provides the DN of the template entry associated with the CoS definition. Used for pointer CoS and classic CoS only.

The `cosAttribute` attribute allows an additional qualifier after the attribute value. You can use the following qualifiers:

- Default

  This qualifier indicates that the server only returns a generated value if there is no corresponding attribute value stored with the entry.

- Override

  This qualifier indicates that the server always returns the value generated by the CoS, even when there is a value stored with the entry.

- Operational

  This qualifier indicates that the attribute will only be returned if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order to be returned. When you use `operational` as a qualifier, it works as if `override` and `operational` were specified.

NOTE You can only make an attribute operational if it is also defined as operational in the schema.

For example, if your CoS generates a value for the `description` attribute, you use the `operational` qualifier because this attribute is not marked operational in the schema.

If you do not indicate a qualifier, `default` is assumed.

For example, you might create a pointer CoS definition entry that contains an `override` qualifier as follows:

```
dn: cn=pointerCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=siroeUS,cn=data
cosAttribute: postalCode override
```

This pointer CoS definition entry indicates that it is associated with a template entry, `cn=siroeUS,cn=data`, that generates the value of the `postalCode` attribute. The override qualifier indicates that this value will take precedence over the value stored by the entries for the `postalCode` attribute.

NOTE	If an entry contains an attribute value generated by a CoS, you cannot manually update the value of the attribute if it is defined with the operational or override qualifiers.

For more information about the attributes, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

Now that you have been introduced to the object classes and attributes used by a CoS definition, it is time to put them together to create the definition entry itself. The following table describes the CoS definition for each type of CoS:

Table 5-3 CoS Definitions

CoS Type	CoS definition

Pointer CoS	`objectclass: top`
	`objectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosPointerDefinition`
	`cosTemplateDn:` *`DN_string`*
	`cosAttribute:` *`list_of_attributes qualifier`*

Indirect CoS	`objectclass: top`
	`objectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosIndirectDefinition`
	`cosIndirectSpecifier:` *`attribute_name`*
	`cosAttribute:` *`list_of_attributes qualifier`*

Table 5-3 CoS Definitions

CoS Type	CoS definition

Classic CoS	`objectclass: top`
	`bbjectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosClassicDefinition`
	`cosTemplateDn:` *`DN_string`*
	`cosSpecifier:` *`attribute_name`*
	`cosAttribute:` *`list_of_attributes qualifier`*

Creating the CoS Template Entry From the Command Line

The CoS template entry also inherits from the `LDAPsubentry` object class. Each template entry is an instance of the `cosTemplate` object class.

NOTE Making the CoS template entry an instance of the `LDAPsubentry` object classes allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else (for example, if it is a user entry), you do not need to make it an instance of the `LDAPsubentry` object class.

The CoS template entry also contains the attribute generated by the CoS (as specified in the `cosAttribute` attribute of the CoS definition entry) and the value for that attribute.

For example, a CoS template entry that provides a value for the `postalCode` attribute follows:

```
dn:cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

It is possible to create CoS templates that compete with each other to provide an attribute value. For example, you might have a multi-valued `cosSpecifier` in your CoS definition entry. In such a case, you can specify a template priority on each template entry to determine which template provides the attribute value. Set the template priority using the `cosPriority` attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

Templates that contain no `cosPriority` attribute are considered the lowest priority. In the case where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.

For example, a CoS template entry for generating a department number appears as follows:

```
dn: cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
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 precedene over any other conflicting templates that define a different `departmentNumber` value.

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

Example of a Pointer CoS

You want to create a pointer CoS that shares a common postal code with all entries in the `dc=siroe,dc=com` tree.

To add a new pointer CoS definition entry to the `dc=siroe,dc=com` suffix, you do an `ldapmodify` as follows:

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

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

Next, you add the pointer CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=pointerCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=siroeUS,cn=data,dc=siroe,dc=com
cosAttribute: postalCode
```

Next, you create the template entry as follows:

```
dn: cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

The CoS template entry (`cn=siroeUS,dn=cata,dc=siroe,dc=com`) supplies the value stored in its `postalCode` attribute to any entries located under the `dc=siroe,dc=com` suffix. These entries are the target entries.

Example of an Indirect CoS

This indirect CoS uses the `team` attribute of the target entry to identify the CoS template entry.

First, you add a new indirect CoS definition entry to the `dc=siroe,dc=com` suffix, using `ldapmodify` as follows:

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

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

Next, you add the indirect CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=indirectCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

Next, you create the template entry for the manager Carla Fuentes as follows:

```
dn:cn=Carla Fuentes,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
departmentNumber: 318842
```

You create a second template entry for the manager Sue Jacobs as follows:

```
dn:cn=Sue Jacobs,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
departmentNumber: 71776
```

The definition entry looks in the target entries (the entries under `dc=siroe,dc=com`) for entries containing the `manager` attribute (because this attribute is specified in the `cosIndirectSpecifier` attribute of the definition entry). The `manager` attribute of the target entry can point to one of two templates, `cn=Carla Fuentes,cn=data,dc=siroe,dc=com` and `cn=Sue Jacobs,cn=data,dc=siroe,dc=com`. The department number is different depending upon the manager.

Example of a Classic CoS

You want to create a classic CoS that automatically generates postal codes using a combination of the template DN and the attribute specified in the `cosSpecifier` attribute.

First, you add a new classic CoS definition entry to the `dc=siroe,dc=com` suffix, using `ldapmodify` as follows:

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

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

Next, you add the indirect CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=classicCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=siroeUS,cn=data,dc=siroe,dc=com
cosSpecifier: employeeType
cosAttribute: postalCode override
```

Next, you create the template entries for the sales and marketing departments as follows:

```
dn: cn=sales,cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

```
dn: cn=marketing,cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 99111
```

The classic CoS definition entry applies to all entries under the `dc=siroe,dc=com` suffix. Depending upon the combination of `employeeType` attribute found in the entry and the cosTemplate DN, it can arrive at one of two templates. One, the sales template, provides a postal code specific to employees in the sales department. The marketing template provides a postal code specific to employees in the marketing department.

Creating Role-Based Attributes

You can create classic CoS schemes that generate attribute values for an entry based on the role possessed by the entry. For example, you could use role-based attributes to set the server look through limit on an entry-by-entry basis.

To create a role-based attribute, use the `nsRole` attribute as the `cosSpecifier` in the CoS definition entry of a classic CoS. Because the `nsRole` attribute can be multivalued, you can define CoS schemes that have more than one possible template entry. To resolve the ambiguity of which template entry to use, you can include the `cosPriority` attribute in your CoS template entry.

For example, you can create a CoS that allows members of the manager role to exceed the standard mailbox quota. The manager role exists as follows:

```
dn: cn=ManagerRole,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: ManagerRole
nsRoleFilter: o=managers
Description: filtered role for managers
```

The classic CoS definition entry would look as follows:

```
dn: cn=managerCOS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectlass: cosClassicDefinition
cosTemplateDn: cn=managerCOS,dc=siroe,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=siroe,dc=com”,cn=managerCOS,
 dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectlass: cosTemplate
mailboxquota: 1000000
```

The template provides the value for the `mailboxquota` attribute, 1000000.

NOTE The role entry and the CoS definition and template entries should be located at the same level in the directory tree.

Access Control and CoS

The server controls access to attributes generated by a CoS in exactly the same way as regular stored attributes. However, access control rules depending upon the value of attributes generated by CoS will not work.

Chapter 6

Managing Access Control

iPlanet Directory Server provides you with the ability to control access to your directory. This chapter describes the access control mechanism.

This section includes the following topics:

- Access Control Principles
- Default ACIs
- Creating ACIs Manually
- Bind Rules
- Creating ACIs From the Console
- Access Control Usage Examples
- Viewing the ACIs for an Entry
- Advanced Access Control: Using Macro ACIs
- Access Control and Replication
- Logging Access Control Information
- Compatibility with Earlier Releases

To take full advantage of the power and flexiblity of the access control mechanism, while you are in the planning phase for your directory deployment, you should define an access control strategy as an integral part of your overall security policy. Refer to *iPlanet Directory Server Deployment Guide* for tips on planning your access control strategy.

Access Control Principles

The mechanism by which you define access is called *access control*. When the server receives a request, it uses the authentication information provided by the user in the bind operation, and the access control instructions (ACIs) defined in the server to allow or deny access to directory information. The server can allow or deny permissions such as read, write, search, and compare. The permission level granted to a user may be dependent on the authentication information provided.

Using access control, you can control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can set permissions for a specific user, all users belonging to a specific group or role, or all users of the directory. Finally, you can define access for a specific location such as an IP address or a DNS name.

ACI Structure

Access control instructions are stored in the directory, as attributes of entries. The `aci` attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by the directory server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The `aci` attribute is returned in an `ldapsearch` operation if specifically requested.

The three main parts of an ACI statement are:

- Target
- Permission
- Bind Rule

The permission and bind rule portions of the ACI are set as a pair, also called an Access Control Rule (ACR). The specified permission is granted or denied depending on whether the accompanying rule is evaluated to be true.

ACI Placement

If an entry containing an ACI does not have any child entries, the ACI applies to that entry only. If the entry has child entries, the ACI applies to the entry itself and all entries below it. As a direct consequence, when the server evaluates access permissions to any given entry, it verifies the ACIs for every entry between the one requested and the directory suffix, as well as the ACIs on the entry itself.

The `aci` attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

You can create an ACI on an entry that does not apply directly to that entry but to some or all of the entries in the subtree below it. The advantage of this is that you can place at a high level in the directory tree a general ACI that effectively applies to entries more likely to be located lower in the tree. For example, at the level of an `organizationalUnit` entry or a `locality` entry, you could create an ACI that targets entries that include the `inetorgperson` object class.

You can use this feature to minimize the number of ACIs in the directory tree by placing general rules at high level branch points. To limit the scope of more specific rules, you should place them as close as possible to leaf entries.

NOTE ACIs placed in the root DSE entry apply only to that entry.

ACI Evaluation

To evaluate the access rights to a particular entry, the server compiles a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored on the directory server. ACIs are evaluated across all of the databases for a particular directory server, but not across directory servers.

The evaluation of this list of ACIs is done based on the semantics of the ACIs, not on their placement in the directory tree. This means that ACIs that are close to the root of the directory tree do not take precedence over ACIs that are closer to the leaves of the directory tree.

The precedence rule that applies is as follows: ACIs that deny access take precedence over ACIs that allow access. Between ACIs that allow access, union semantics apply, so there is no precedence.

For example, if you deny write permission at the directory's root level, then none of the users can write to the directory regardless of the specific permissions you grant them. To grant a specific user write permissions to the directory, you have to restrict the scope of the original denial for write permission so that it does not include the user.

ACI Limitations

When creating an access control policy for your directory service, you need to be aware of the following restrictions:

- If your directory tree is distributed over several servers using the chaining feature, some restrictions apply to the keywords you can use in access control statements:
  - ACIs that depend on group entries (`groupdn` keyword) must be located on the same server as the group entry. If the group is dynamic, then all members of the group must have an entry on the server too. If the group is static, the members's entries can be located on remote servers.
  - ACIs that depend on role definitions (`roledn` keyword) must be located on the same server as the role definition entry. Every entry that is intended to have the role must also be located on the same server.

  However, you can do value matching of values stored in the target entry with values stored in the entry of the bind user (for example, using the userattr keyword). Access will be evaluated normally even if the bind user does not have an entry on server that holds the ACI.

  For more information on how to chain access control evaluation, see "Database Links and Access Control Evaluation," on page 106.
- Attributes generated by a CoS cannot be used in all ACI keywords.Specifically, you should not use attributes generated by CoS with the following keywords:
  - `targetfilter` (see "Targeting Entries or Attributes Using LDAP Filters," on page 193)
  - `targattrfilters` (see "Targeting Attribute Values Using LDAP Filters," on page 194)
  - `userattr` (see "Using the userattr Keyword," on page 208)

  If you create target filters or bind rules that depend on the value of attributes generated by CoS, the access control rule will not work. For more information on CoS, see Chapter 5, "Advanced Entry Management."

- Access control rules are always evaluated on the local server. Therefore, it is not necessary to specify the hostname or port number of the server in LDAP URLs used in ACI keywords. If you do, the LDAP URL will not be taken into account at all. For more information on LDAP URLs, see Appendix C, “LDAP URLs.”

Default ACIs

When you install the directory server, the following default ACIs apply to your directory information stored in the `userRoot` database:

- Users can modify their own entry in the directory, but not delete it. They cannot modify the `aci` and `nsroledn` attributes.
- Users have anonymous access to the directory for search, compare, and read operations.
- The administrator (by default `uid=admin,ou=Administrators, ou=TopologyManagement,o=NetscapeRoot`) has all rights except proxy rights.
- All members of the Configuration Administrators group have all rights except proxy rights.
- All members of the Directory Administrators group have all rights except proxy rights.
- SIE group.

Whenever you create a new database in the directory, the top entry has the default ACIs listed above.

The NetscapeRoot subtree has its own set of default ACIs:

- All members of the Configuration Administrators group have all rights on the NetscapeRoot subtree except proxy rights.
- Users have anonymous access to the NetscapeRoot subtree for search and read operations.
- Group expansion.
- All authenticated users have search, compare, and read rights to configuration attributes that identify the administration server.

The following sections explain how to modify these default settings to suit the needs of your organization.

Creating ACIs Manually

You can create access control instructions manually using LDIF statements, and add them to your directory tree using the `ldapmodify` utility. The following sections explain in detail how to create the LDIF statements.

TIP LDIF ACI statements can be very complex. However, if you are setting access control for a large number of directory entries, using LDIF is the preferred method over using the Console because of the time it can save.

To familiarize yourself with LDIF ACI statements, however, you may want to use the Directory Server Console to set the ACI and then click the Edit Manually button on the Access Control Editor. This shows you the correct LDIF syntax. If your operating system allows it, you can even copy the LDIF from the Access Control Editor and paste it into your LDIF file.

The ACI Syntax

The `aci` attribute uses the following syntax:

`aci: (`*target*`)(version 3.0;acl "`*name*`";`*permission bind_rules*`;)`

where

- *target* specifies the entry, attributes, or set of entries and attributes for which you want to control access. The target can be a distinguished name, one or more attributes, or a single LDAP filter. The target is an optional part of the ACI.
- `version 3.0` is a required string that identifies the ACI version.
- "*name*" is a name for the ACI. The name can be any string that identifies the ACI. The ACI name is required.
- *permission* specifically outlines what rights you are either allowing or denying (for example, read or search rights).
- *bind_rules* specify the credentials and bind parameters that a user has to provide to be granted access. Bind rules can also specifically deny access to certain users or groups of users.

You can have multiple permission-bind rule pairs for each target. This allows you to efficiently set multiple access controls for a given target. For example:

target`(`*permission bind_rule*`)(`*permission bind_rule*`)...`

If you have several ACRs in one ACI statement, the syntax is of the form:

`aci: (`*target*`)(version 3.0;acl "`*name*`";`*permission bind_rule*`;` *permission bind_rule*`;` `...` *permission bind_rule*`;)`

Example ACI

The following is an example of a complete LDIF ACI:

```
aci: (target="ldap:///uid=bjensen,dc=siroe,dc=com")(targetattr=*)
(version 3.0;acl "aci1";allow (write) userdn="ldap:///self";)
```

In this example, the ACI states that the user bjensen has rights to modify all attributes in her own directory entry.

The following sections describe the syntax of each portion of the ACI in more detail.

Defining Targets

The target identifies what the ACI applies to. If the target is not specified, the ACI applies to the entry containing the `aci` attribute and to the entries below it.

A target can be:

- A directory entry, or all of the entries in a subtree, as described in “Targeting a Directory Entry,” on page 190.
- Attributes of an entry, as described in “Targeting Attributes,” on page 192.
- A set of entries or attributes that match a specified LDAP filter, as described in “Targeting Entries or Attributes Using LDAP Filters,” on page 193.
- An attribute value, or a combination of values, that match a specified LDAP filter, as described in “Targeting Attribute Values Using LDAP Filters,” on page 194.

The general syntax for a target is:

`(`*keyword* `= "`*expression*`")`

`(`*keyword* `!= "`*expression*`")`

where:

  - *keyword* indicates the type of target

- equal (=) indicates that the target is the object specified in the *expression*, and not equal (!=) indicates the target is not the object specified in the *expression*.
- *expression* identifies the target

The quotation marks ("") around *expression* are required. What you use for *expression* is dependent upon the *keyword* that you supply.

The following table lists each keyword and the associated expressions:

Table 6-1 LDIF Target Keywords

Keyword	Valid Expressions	Wildcard Allowed?
`target`	ldap:///*distinguished_name*	yes
`targetattr`	*attribute*	yes
`targetfilter`	*LDAP_filter*	yes
`targattrfilters`	*LDAP_operation:LDAP_filter*	yes

In all cases, you must keep in mind that when you place an ACI on an entry, if it is not a leaf entry, the ACI also applies to all entries below it. For example, if you target the entry `ou=accounting,dc=siroe,dc=com`, the permissions you set will apply to all entries in the accounting branch of the Siroe tree.

As a counter example, if you place an ACI on the `ou=accounting,dc=siroe,dc=com` entry, you cannot target the `uid=sarette,ou=people,dc=siroe,dc=com` entry because it is not located under the accounting tree.

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=siroe,dc=com")
```

NOTE If the DN of the entry to which the access control rule applies contains a comma, you must escape the comma with a single backslash (\). For example:

```
(target="ldap:///uid=lfuentes,dc=Siroe Bolivia\,S.A.")
```

You can also use a wildcard when targeting a distinguished name using the `target` keyword. The wildcard indicates that any character or string or substring is a match for the wildcard. Pattern matching is based on any other strings that have been specified with the wildcard.

The following are legal examples of wildcard usage:

- `(target="ldap:///uid=*,dc=siroe,dc=com")`

  Matches every entry in the entire Siroe tree that has the `uid` attribute in the entry's RDN.

- `(target="ldap:///uid=*Anderson,dc=siroe,dc=com")`

  Matches every entry directly under the Siroe node with a `uid` ending in Anderson.

- `(target="ldap:///uid=C*A,dc=siroe,dc=com")`

  Matches every entry directly under the Siroe node with a `uid` beginning with C and ending with A.

Depending on the position of the wildcard, it can apply to the full DN, not only to attribute values. Therefore, the wildcard can be used as a substitute for portions of the DN. For example, `uid=andy*,dc=siroe,dc=com` targets all the directory entries in the entire Siroe tree with a matching uid attribute, and not just the entries that are immediately below the `dc=siroe,dc=com` node. In other words, this target matches with longer expressions such as `uid=andy,ou=eng,dc=siroe,dc=com`, or `uid=andy,ou=marketing,dc=siroe,dc=com`.

Some other valid examples follow:

- `(target="ldap:///uid=*,dc=siroe,dc=com")`

  Matches every entry in the entire Siroe tree that has the `uid` attribute in the entry's RDN.

- `(target="ldap:///uid=*,ou=*,dc=siroe,dc=com")`

  Matches every entry in the Siroe tree whose distinguished name contains the `uid` and `ou` attributes. Thus:

```
uid=fchen,ou=Engineering,dc=siroe,dc=com
```

or

```
uid=claire,ou=Engineering,ou=people,dc=siroe,dc=com
```

would match, but the following would not:

```
uid=bjensen,dc=siroe,dc=com
ou=Engineering,dc=siroe,dc=com
```

NOTE You cannot use wildcards in the suffix part of a distinguished name. That is, if your directory uses the suffixes `c=US` and `c=GB`, then you *cannot* use the following target to reference both suffixes:

```
(target="ldap:///dc=siroe,c=*").
```

Neither can you use a target such as `uid=bjensen,dc=*.com`.

Targeting Attributes

In addition to targeting directory entries, you can also target one or more attributes included in the targeted entries. This is useful when you want to deny or allow access to partial information about an entry. For example, you could allow access to only the common name, surname, and telephone number attributes of a given entry. Or you could deny access to sensitive information such as passwords.

You can specify that the target is equal or is not equal to a specific attribute. The attributes you supply do not need to be defined in the schema. This absence of schema checking makes it possible to implement an access control policy when you set up your directory service for the first time, even if the ACLs you create do not apply to the current directory content.

To target attributes, you use the `targetattr` keyword. The `targetattr` keyword uses the following syntax:

`(targetattr = "`*attribute*`")`

You can target multiple attributes by using the `targetattr` keyword with the following syntax:

`(targetattr = "`*attribute1* `||` *attribute2 ...* `||` *attributen*`")`

Where *attribute* is the name of the attribute you want to target.

For example, to target the common name attribute you would use:

```
(targetattr = "cn")
```

To target an entry's common name, surname, and uid attributes, you would use the following:

```
(targetattr = "cn || sn || uid")
```

The attributes specified in the `targetattr` keyword apply to the entry that the ACI is targeting, and to all the entries below it. That is, if you target the password attribute on the entry `uid=bjensen,ou=Marketing,dc=siroe,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=siroe,dc=com`, then all the entries beneath the branch point that can contain a password attribute are affected by the ACI.

Targeting Both an Entry and Attributes

By default, the entry targeted by an ACI containing a `targetattr` keyword is the entry on which the ACI is placed. That is, if you put the ACI

`aci: (targetattr = "uid")(`*access_control_rules*`;)`

on the `ou=Marketing, dc=siroe,dc=com` entry, then the ACI applies to the entire Marketing subtree. However, you can also explicitly specify a target using the `target` keyword as follows:

`aci: (target="ldap:///ou=Marketing,`
`dc=siroe,dc=com")(targetattr="uid") (`*access_control_rules*`;)`

The order in which you specify the `target` and the `targetattr` keywords is not important.

Targeting Entries or Attributes Using LDAP Filters

You can use LDAP filters to target a group of entries that match certain criteria. To do this, you must use the `targetfilter` keyword with an LDAP filter.

The syntax of the `targetfilter` keyword is:

`(targetfilter = "`*LDAP_filter*`")`

where *LDAP_filter* is a standard LDAP search filter. For more information on the syntax of LDAP search filters, see Appendix B, "Finding Directory Entries."

For example, suppose that all entries in the accounting department include the attribute- value pair `ou=accounting`, and all entries in the engineering department include the attribute- value pair `ou=engineering` subtree. To target all the entries in the accounting and engineering branches of the directory tree, you could use the following filter:

```
(targetfilter = "(|(ou=accounting)(ou=engineering))")
```

This type of filter targets whole entries. You can associate the `targetfilter` and the `targetattr` keywords to create ACIs that apply to a subset of attributes in the targeted entries.

The following LDIF example allows members of the Engineering Admins group to modify the `departmentNumber` and `manager` attributes of all entries in the Engineering business category. This example uses LDAP filtering to select all entries with `businessCategory` attributes set to Engineering:

```
dn: dc=siroe,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=siroe,dc=com";)
```

TIP Although using LDAP filters can be useful when you are targeting entries and attributes that are spread across the directory, the results are sometimes unpredictable because filters do not directly name the object for which you are managing access. The set of entries targeted by a filtered ACI is likely to change as attributes are added or deleted. Therefore, if you use LDAP filters in ACIs, you should verify that they target the correct entries and attributes by using the same filter in an `ldapsearch` operation.

Targeting Attribute Values Using LDAP Filters

You can use access control to target specific attribute values. This means that you can grant or deny permissions on an attribute if that attribute's value meets the criteria defined in the ACI. An ACI that grants or denies access based on an attribute's value, is called a value-based ACI.

For example, you might grant all users in your organization permission to modify the `nsRoleDN` attribute in their own entry. However, you would also want to ensure that they do not give themselves certain key roles such as “Top Level Administrator.” LDAP filters are used to check that the conditions on attribute values are satisfied.

To create a value based ACI, you must use the `targattrfilters` keyword with the following syntax:

`(targattrfilters=”add=`*attr1*`:`*F1* `&&` *attr2*`:`*F2*`... &&` *attrn*`:`*Fn*`,del=`*attr1*`:`*F1* `&&` *attr2*`:`*F2* `... &&` *attrn*`:`*Fn*`”)`

where:

- `add` represents the operation of creating an attribute
- `del` represents the operation of deleting an attribute
- *attrx* represents the target attributes
- *Fx* represents filters that apply only to the associated attribute

When creating an entry, if a filter applies to an attribute in the new entry, then each instance of that attribute must satisfy the filter. When deleting an entry, if a filter applies to an attribute in the entry, then each instance of that attribute must also satisfy the filter.

When modifying an entry, if the operation adds an attribute, then the add filter that applies to that attribute must be satisfied; if the operation deletes an attribute, then the delete filter that applies to that attribute must be satisfied. If individual values of an attribute already present in the entry are replaced, then both the add and delete filters must be satisfied.

For example consider the following attribute filter:

```
(targattrfilters=”add=nsroleDN:(!(nsRoleDN=cn=superAdmin)) &&
telephoneNumber:(telephoneNumber=123*))
```

This filter can be used to allow users to add any role (`nsRoleDN` attribute) to their own entry, except the `superAdmin` role. It also allows users to add a telephone number with a 123 prefix.

NOTE You cannot create value-based ACIs from the Server Console.

Targeting a Single Directory Entry

Targeting a single directory entry is not straightforward because it goes against the design philosophy of the access control mechanism. However, it can be done:

- By creating a bind rule that matches user input in the bind request with an attribute value stored in the targeted entry. For more details, see "Defining Access Based on Value Matching," on page 207.
- 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=siroe,dc=com`, and there aren't any organizational units (`ou`) defined below that node you could specify an ACI that contains:

```
targetattr=ou
```

A safer method is to use the `targetfilter` keyword and to explicitly specify 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 methods is that your directory tree might change in the future, and you would have to remember to modify this ACI.

Defining Permissions

Permissions specify the type of access you are allowing or denying. You can either allow or deny permission to perform specific operations in the directory. The various operations that can be assigned are known as *rights*.

There are two parts to setting permissions:

- Allowing or denying access
- Assigning rights

Allowing or Denying Access

You can either explicitly allow or deny access permissions to your directory tree. For more guidelines on when to allow and when to deny access, refer to the *iPlanet Directory Server Deployment Guide*.

NOTE From the Server Console, you cannot explicitly deny access, but only grant permissions.

Assigning Rights

Rights detail the specific operations a user can perform on directory data. You can allow or deny all rights, or you can assign one or more of the following rights:

Read. Indicates whether users can read directory data. This permission applies only to the search operation.

Write. Indicates whether users can modify an entry by adding, modifying, or deleting *attributes*. This permission applies to the modify and modrdn operations.

Add. Indicates whether users can create *entries*. This permission applies only to the add operation.

Delete. Indicates whether users can delete *entries*. This permission applies only to the delete operation.

Search. Indicates whether users can search for the directory data. Users must have Search and Read rights in order to view the data returned as part of a search result. This permission applies only to the search operation.

Compare. Indicates whether the users can compare data they supply with data stored in the directory. With compare rights, the directory returns a success or failure message in response to an inquiry, but the user cannot see the value of the entry or attribute. This permission applies only to the compare operation.

Selfwrite. Indicates whether users can add or delete their own DN from a group. This right is used only for group management.

Proxy. Indicates whether the specified DN can access the target with the rights of another entry. For an overview of proxy access, refer to *iPlanet Directory Server Deployment Guide*.

All. Indicates that the specified DN has all rights (read, write, search, delete, compare, and selfwrite) to the targeted entry, *excluding* proxy rights.

Rights are granted independently of one another. This means, for example, that a user who is granted add rights can create an entry but cannot delete it if delete rights have not been specifically granted. Therefore, when planning the access control policy for your directory, you must ensure that you grant rights in a way that makes sense for users. For example, it doesn't usually make sense to grant write permission without granting read and search permissions.

Rights Required for LDAP Operations

This section describes the rights you need to grant to users depending on the type of LDAP operation you want to authorize them to perform.

Adding an entry:

- Grant add permission on the entry being added.
- Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.

Deleting an entry:

- Grant delete permission on the entry to be deleted.
- Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the `targattrfilters` keyword.

Modifying an attribute in an entry:

- Grant write permission on the attribute type.
- Grant write permission on the value of each attribute type. This right is granted by default but could be restricted using the `targattrfilters` keyword.

Modifying the RDN of an entry:

- Grant write permission on the entry.
- Grant write permission on the attribute type used in the new RDN.
- Grant write permission on the attribute type used in the old RDN, if you want to grant the right to delete the old RDN.
- Grant write permission on the value of attribute type used in the new RDN. This right is granted by default but could be restricted using the `targattrfilters` keyword.

Comparing the value of an attribute:

- Grant compare permission on the attribute type.

Searching for entries:

- Grant search permission on each attribute type used in the search filter.
- Grant read permission on attribute types used in the entry.

The permissions you need to set up to allow users to search the directory are more readily understood with an example. Consider the following `ldapsearch` operation:

```
% ldapsearch -h host -s base -b "uid=bkolics,dc=siroe,dc=com" objectclass=* mail
```

The following ACI is used to determine whether user `bkolics` can be granted access:

```
aci: (targetattr = "mail")(version 3.0; acl "self access to mail";
allow (read, search) userdn = "ldap:///self";)
```

The search result list is empty, because this ACI does not grant access to the objectclass attribute. If you want the search operation described above to be successful, you must modify the ACI to read as follows:

```
aci: (targetattr = "mail || objectclass")(version 3.0; acl "self
access to mail"; allow (read, search) userdn = "ldap:///self";)
```

Permissions Syntax

In an ACI statement, the syntax for permissions is:

`allow|deny (`*rights*`)`

where *rights* is a list of 1 to 8 comma-separated keywords enclosed within parentheses. Valid keywords are **`read`**, **`write`**, **`add`**, **`delete`**, **`search`**, **`compare`**, **`selfwrite`**, **`proxy`**, or **`all`**.

In the following example, read, search, and compare access is allowed, provided the bind rule is evaluated to be true:

```
aci:  (target="ldap:///dc=siroe,dc=com") (version 3.0;acl "example";
allow (read, search, compare) bind_rule;)
```

Bind Rules

Depending on the ACIs defined for the directory, for certain operations, you need to *bind* to the directory. *Binding* means logging in or authenticating yourself to the directory by providing a bind DN and password, or, if using SSL, a certificate. The credentials provided in the bind operation, and the circumstances of the bind determine whether access to the directory is allowed or denied.

Every permission set in an ACI has a corresponding bind rule that details the required credentials and bind parameters.

Bind rules can be simple. For example, a bind rule can simply state that the person accessing the directory must belong to a specific group. Bind rules can also be more complex. For example, a bind rule can state that a person must belong to a specific group and must log in from a machine with a specific IP address, between 8 am and 5 pm.

Bind rules define who can access the directory, when, and from where. More specifically, bind rules can specify:

- Users, groups, and roles that are granted access
- Location from which an entity must bind
- Time or day on which binding must occur
- Type of authentication that must be in use during binding

Additionally, bind rules can be complex constructions that combine these criteria by using Boolean operators. See "Using Boolean Bind Rules," on page 217 for more information.

Bind Rule Syntax

Whether access is allowed or denied depends on whether an ACI's bind rule is evaluated to be true. Bind rules use one of the two following patterns:

keyword `= "`*expression*`";`

keyword `!= "`*expression*`";`

where equal (=) indicates that *keyword* and *expression* must match in order for the bind rule to be true, and not equal (!=) indicates that *keyword* and *expression* must not match in order for the bind rule to be true.

NOTE The timeofday keyword also supports the inequality expressions (<, <=, >, >=). This is the only keyword that supports these expressions.

The quotation marks ("") around *expression* and the delimiting semicolon (;) are required. The expressions you can use depend on the associated *keyword*.

The following table lists each keyword and the associated expressions. It also indicates whether wildcard characters are allowed in the expression.

Table 6-2 LDIF Bind Rule Keywords

Keyword	Valid Expressions	Wildcard Allowed?
`userdn`	ldap:///*distinguished_name*	yes, in DN only
	ldap:///all	
	ldap:///anyone	
	ldap:///self	
	ldap:///parent	
	ldap:///*suffix*??sub?(*filter*)	
`groupdn`	ldap:///*DN* || *DN*	no
`roledn`	ldap:///*DN* || *DN*	no
`userattr`	*attribute#bindType* or	no
	attribute#value	
`ip`	*IP_address*	yes
`dns`	*DNS_host_name*	yes
`dayofweek`	sun	no
	mon	
	tue	
	wed	
	thu	
	fri	
	sat	
`timeofday`	0 - 2359	no

Table 6-2 LDIF Bind Rule Keywords *(Continued)*

Keyword	Valid Expressions	Wildcard Allowed?
`authmethod`	none simple ssl sasl *authentication_method*	no

The following sections further detail the bind rule syntax for each keyword.

Defining User Access - userdn Keyword

User access is defined using the `userdn` keyword. The `userdn` keyword requires one or more valid distinguished names in the following format :

`userdn = "ldap:///`*`dn`*` [|| ldap:///`*`dn`*`]...[||ldap:///`*`dn`*`]"`

where *dn* can be a DN or one of the expressions **anyone**, **`all`**, **`self`** or **`parent`**:

`userdn = "ldap:///anyone"` - defines anonymous access

`userdn = "ldap:///all"` - defines general access

`userdn = "ldap:///self"` - defines self access

`userdn = "ldap:///parent"` - defines access for the parent entry

The `userdn` keyword can also be expressed as an LDAP filter of the form:

`ldap:///`*`suffix`*`??sub?(`*`filter`*`)`

NOTE	If a DN contains a comma, the comma must be preceded by a backslash (\) escape character.

Anonymous Access (anyone Keyword)

Granting anonymous access to the directory means that anyone can access it without providing a bind DN or password, and regardless of the circumstances of the bind. You can limit anonymous access to specific types of access (for example, access for read or access for search) or to specific subtrees or individual entries within the directory.

From the Server Console, you define anonymous access through the Access Control Editor. See “Creating ACIs From the Console,” on page 218.

General Access (all Keyword)

You can use bind rules to indicate that a permission applies to anyone who has successfully bound to the directory; that is, all authenticated users. This allows general access while preventing anonymous access.

From the Server Console, you define general access on the Access Control Editor. For more information, see “Creating ACIs From the Console,” on page 218.

Self Access (self Keyword)

Specifies that users are granted or denied access to their own entries. In this case, access is granted or denied if the bind DN matches the DN of the targeted entry.

From the Server Console, you set up self access on the Access Control Editor. For more information, see “Creating ACIs From the Console,” on page 218.

Parent Access (parent Keyword)

Specifies that users are granted or denied access to the entry only if their bind DN is the parent of the targeted entry.

You cannot set up parent access control using the Server Console.

LDAP URLs

You can dynamically target users in ACIs using a URL with a filter as follows:

```
userdn = "ldap:///<suffix>??sub?(filter)"
```

For example, all users in the accounting and engineering branches of the Siroe tree would be granted or denied access to the targeted resource dynamically based on the following URL:

```
userdn = "ldap:///dc=siroe,dc=com??sub?(|(ou=engineering)(ou=accounting))"
```

NOTE Do not specify a hostname or port number within the LDAP URL. LDAP URLs always apply to the local server.

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

Wildcards

You can also specify a set of users by using the wildcard character (*). For example, specifying a user DN of `uid=u*,dc=siroe,dc=com` indicates that only users with a bind DN beginning with the letter `u` will be allowed or denied access based on the permissions you set.

From the Server Console, you set user access from the Access Control Editor. For more information, see "Creating ACIs From the Console," on page 218.

Examples

This section contains examples of the `userdn` syntax.

Userdn keyword containing an LDAP URL:

```
userdn = "ldap:///uid=*,dc=siroe,dc=com";
```

The bind rule is evaluated to be true if the user binds to the directory using any distinguished name of the specified pattern. For example, both of the following bind DNs would be evaluated to be true:

```
uid=ssarette,dc=siroe,dc=com
uid=tjaz,ou=Accounting,dc=siroe,dc=com
```

whereas the following bind DN would be evaluated to be false:

```
cn=Babs Jensen,dc=siroe,dc=com
```

Userdn keyword containing logical OR of LDAP URLs:

```
userdn="ldap:///uid=bj,c=siroe.com ||
ldap:///uid=kc,dc=siroe,dc=com";
```

The bind rule is evaluated to be true if the client binds as either of the two supplied distinguished names.

Userdn keyword excluding a specific LDAP URL:

```
userdn != "ldap:///uid=*,ou=Accounting,dc=siroe,dc=com";
```

The bind rule is evaluated to be true if the client is not binding as a UID-based distinguished name in the accounting subtree. This bind rule only makes sense if the targeted entry is not under the accounting branch of the directory tree.

Userdn keyword containing self keyword:

```
userdn = "ldap:///self";
```

The bind rule is evaluated to be true if the user is accessing the entry represented by the DN with which the user bound to the directory. That is, if the user has bound as `uid=ssarette, dc=siroe,dc=com` and the user is attempting an operation on the `uid=ssarette,dc=siroe,dc=com` entry, then the bind rule is true.

For example, if you want to grant all users in the Siroe tree write access to their `userPassword` attribute, you would create the following ACI on the `dc=siroe,dc=com` node.

```
aci: (targetattr = "userPassword") (version 3.0; acl "write-self";
allow (write) userdn = "ldap:///self";)
```

Userdn keyword containing the all keyword:

```
userdn = "ldap:///all";
```

The bind rule is evaluated to be true for any valid bind DN. To be true, a valid distinguished name and password must have been presented by the user during the bind operation.

For example, if you want to grant read access to the entire tree to all authenticated users, you would create the following ACI on the `dc=siroe,dc=com` node:

```
aci:(version 3.0; acl "all-read"; allow (read)
userdn="ldap:///all";)
```

Userdn keyword containing the anyone keyword:

```
userdn = "ldap:///anyone";
```

The bind rule is evaluated to be true for anyone; use this keyword to provide anonymous access to your directory.

For example, if you want to allow anonymous read and search access to the entire Siroe tree, you would create the following ACI on the `dc=siroe,dc=com` node:

```
aci: (version 3.0; acl "anonymous-read-search"; allow (read, search)
userdn = "ldap:///anyone";)
```

Userdn keyword containing the parent keyword:

```
userdn = "ldap:///parent";
```

The bind rule is evaluated to be true if the bind DN is the parent of the targeted entry.

For example, if you want to grant write access to every user's child entries, you would create the following ACI on the `dc=siroe,dc=com` node:

```
aci:(version 3.0; acl "parent access"; allow (write)
userdn="ldap:///parent";)
```

```
userdn = "ldap:///dc=siroe,dc=com???(|(ou=engineering)(ou=sales))";
```

The bind rule is evaluated to be true if the user belongs to the engineering or sales subtree.

Defining Group Access - groupdn Keyword

Members of a specific group can access a targeted resource. This is known as *group access*. Group access is defined using the `groupdn` keyword to specify that access to a targeted entry will be granted or denied if the user binds using a DN that belongs to a specific group.

The `groupdn` keyword requires one or more valid distinguished names in the following format :

```
groupdn="ldap:///dn [|| ldap:///dn]...[|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the named group.

NOTE If a DN contains a comma, the comma must be escaped by a backslash (\).

From the Server Console, you can define specific groups using the Access Control Editor. For more information, see "Creating ACIs From the Console," on page 218.

Examples

This section contains examples of the `groupdn` syntax.

Groupdn keyword containing an LDAP URL:

```
groupdn = "ldap:///cn=Administrators,dc=siroe,dc=com";
```

The bind rule is evaluated to be true if the bind DN belongs to the Administrators group. If you wanted to grant the Administrators group permission to write to the entire directory tree, you would create the following ACI on the `dc=siroe,dc=com` node:

```
aci: (version 3.0; acl "Administrators-write"; allow (write)
groupdn="ldap:///cn=Administrators,dc=siroe,dc=com";)
```

Groupdn keyword containing logical OR of LDAP URLs:

```
groupdn = "ldap:///cn=Administrators,dc=siroe,dc=com" ||
"ldap:///cn=Mail Administrators,dc=siroe,dc=com";
```

The bind rule is evaluated to be true if the bind DN belongs to either the Administrators or the Mail Administrators group.

Defining Role Access - roledn Keyword

Members of a specific role can access a targeted resource. This is known as *role access*. Role access is defined using the `roledn` keyword to specify that access to a targeted entry will be granted or denied if the user binds using a DN that belongs to a specific role.

The `roledn` keyword requires one or more valid distinguished names in the following format :

```
roledn = "ldap:///dn [|| ldap:///dn]... [|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the specified role.

NOTE	If a DN contains a comma, the comma must be escaped by a backslash (\).

The `roledn` keyword has the same syntax and is used in the same way as the `groupdn` keyword.

Defining Access Based on Value Matching

You can set bind rules to specify that an attribute value of the entry used to bind to the directory must match an attribute value of the targeted entry.

For example, you can specify that the bind DN must match the DN in the `manager` attribute of a user entry in order for the ACI to apply. In this case, only the user's manager would have access to the entry.

This example is based on DN matching. However, you can match any attribute of the entry used in the bind with the targeted entry. For example, you could create an ACI that allowed any user whose `favoriteDrink` attribute is "beer" to read all the entries of other users that have the same value for `favoriteDrink`.

Using the userattr Keyword

The `userattr` keyword can be used to specify which attribute values must match between the entry used to bind and the targeted entry. You can specify:

- A user DN
- A group DN
- A role DN
- An LDAP filter, in an LDAP URL
- Any attribute type

The LDIF syntax of the `userattr` keyword is as follows:

`userattr = "`*attrName#bindType*`"`

or, if you are using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter:

`userattr = "`*attrName#attrValue*`"`

where:

- *attrName* is the name of the attribute used for value matching
- *bindType* is one of `USERDN, GROUPDN, ROLEDN, LDAPURL`
- *attrValue* is any string representing an attribute value

The following sections provide examples of the `userattr` keyword with the various possible bind types.

Example with USERDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on the user DN:

```
userattr = "manager#USERDN"
```

The bind rule is evaluated to be true if the bind DN matches the value of the `manager` attribute in the targeted entry. You can use this to allow a user's manager to modify employees' attributes. This mechanism only works if the `manager` attribute in the targeted entry is expressed as a full DN.

The following example grants a manager full access to his or her employees' entries:

```
aci: (target="ldap:///dc=siroe,dc=com")(targetattr=*) (version 3.0;
acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

Example with GROUPDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on a group DN:

```
userattr = "owner#GROUPDN"
```

The bind rule is evaluated to be true if the bind DN is a member of the group specified in the `owner` attribute of the targeted entry. For example, you can use this mechanism to allow a group to manage employees' status information. You can use an attribute other than `owner`, as long as the attribute you use contains the DN of a group entry.

The group you point to can be a dynamic group, and the DN of the group can be under any suffix in the database. However, the evaluation of this type of ACI by the server is very resource intensive.

If you are using static groups that are under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=siroe,dc=com?owner#GROUPDN"
```

In this example, the group entry is under the `dc=siroe,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

Example With ROLEDN Bind Type

The following is an example of the `userattr` keyword associated with a bind based on a role DN:

```
userattr = "siroeEmployeeReportsTo#ROLEDN"
```

The bind rule is evaluated to be true if the bind DN belongs to the role specified in the `siroeEmployeeReportsTo` 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 `siroeEmployeeReportsTo` attribute to the schema, and that all employee entries contain this attribute. It also assumes that the value of this attribute is the DN of a role entry.

For information on designing your schema, refer to *iPlanet Directory Server Deployment Guide*. For information on adding attributes to the schema, see "Creating Attributes," on page 320.

The DN of the role can be under any suffix in the database. If, in addition, you are using filtered roles, the evaluation of this type of ACI uses a lot of resources on the server.

If you are using a static role definition, and the role entry is under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=siroe,dc=com?employeeReportsTo#ROLEDN"
```

In this example, the role entry is under the `dc=siroe,dc=com` suffix. The server can process this type of syntax more quickly than the previous example.

Example With LDAPURL Bind Type

The following is an example of the `userattr` keyword associated with a bind based on an LDAP filter:

userattr = "*myfilter*#LDAPURL"

The bind rule is evaluated to be true if the bind DN matches the filter specified in the *myfilter* attribute of the targeted entry. The *myfilter* attribute can be replaced by any attribute that contains an LDAP filter.

Example With Any Attribute Value

The following is an example of the `userattr` keyword associated with a bind based on any attribute value:

```
userattr = "favoriteDrink#Beer"
```

The bind rule is evaluated to be true if the bind DN and the target DN include the `favoriteDrink` attribute with a value of **`Beer`**.

Using the userattr Keyword With Inheritance

When you use the `userattr` keyword to associate the entry used to bind with the target entry, the ACI applies only to the target specified and not to the entries below it. In some circumstances, you might want to extend the application of the ACI several levels below the targeted entry. This is possible by using the parent keyword, and specifying the number of levels below the target that should inherit the ACI.

When you use the `userattr` keyword in association with the `parent` keyword, the syntax is as follows:

userattr = "parent[*inheritance_level*].*attrName*#*bindType*"

or, if you are using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter:

userattr = "parent[*inheritance_level*].*attrName*#*attrValue*" where :

- *inheritance_level* is a comma separated list that indicates how many levels below the target will inherit the ACI. You can include five levels `[0,1,2,3,4]` below the targeted entry; zero (0) indicates the targeted entry.
- *attribute* is the attribute targeted by the `userattr` or `groupattr` keyword.
- *bindType* can be one of `USERDN, GROUPDN, ROLEDN, LDAPURL`.

For example,

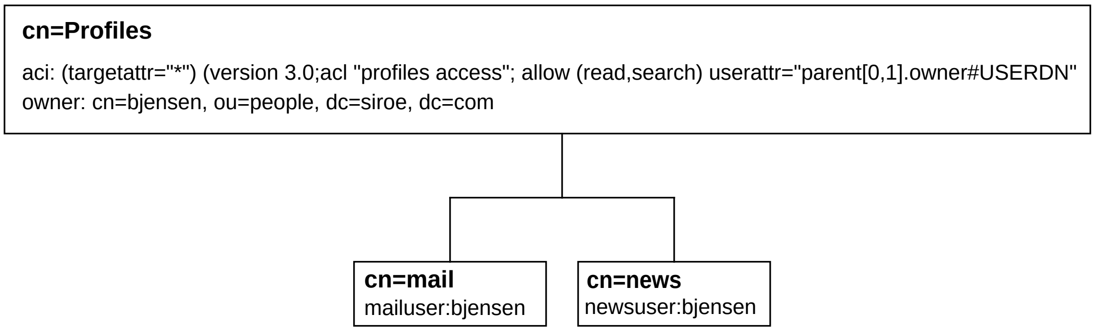
```
userattr = "parent[0,1].manager#USERDN"
```

This bind rule is evaluated to be true if the bindDN matches the manager attribute of the targeted entry. The permissions granted when the bind rule is evaluated to be true apply to the target entry *and* to all entries immediately below it.

Example With userattr Inheritance

The example in the following figure indicates that user `bjensen` is allowed to read and search the `cn=Profiles` entry as well as the first level of child entries which includes `cn=mail` and `cn=news`, thus allowing her to search through her own mail and news IDs.

Figure 6-1 Using Inheritance With the `userattr` Keyword


cn=Profiles
aci: (targetattr="*") (version 3.0;acl "profiles access"; allow (read,search) userattr="parent[0,1].owner#USERDN"
owner: cn=bjensen, ou=people, dc=siroe, dc=com
cn=mail
mailuser:bjensen
cn=news
newsuser:bjensen

In this example, if you did not use inheritance you would have to do one of the following to achieve the same result:

- Explicitly set read and search access for user `bjensen` on the `cn=Profiles`, `cn=mail`, and `cn=news` entries in the directory.
- Add the owner attribute with a value of `bjensen` to the `cn=mail` and `cn=news` entries and then add the following ACI to the `cn=mail` and `cn=news` entries.

```
   aci: (targetattr="*") (version 3.0; acl "profiles access"; allow
   (read,search) userattr="owner#USERDN";)
```

Granting Add Permission Using the userattr Keyword

If you use the `userattr` keyword in conjunction with **`all`** or **`add`** permissions, you might find that the behavior of the server is not what you expect. Typically, when a new entry is created in the directory, Directory Server evaluates access rights on the entry being created, and not on the parent entry. However, in the case of ACIs using the `userattr` keyword, this behavior could create a security hole, and the server's normal behavior is modified to avoid it.

Consider the following example:

```
aci: (target="ldap:///dc=siroe,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=siroe,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=siroe,dc=com

objectclass: top

...

cn: Trojan Horse

manager: cn=Joe,ou=eng,dc=siroe,dc=com
```

To avoid this type of security threat, the ACI evaluation process does not grant add permission at level 0, that is, 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=siroe,dc=com` that has a `manager` attribute that matches the bind DN:

```
aci: (target="ldap:///dc=siroe,dc=com")(targetattr=*)
(version 3.0; acl "parent-access"; allow (add)
userattr = "parent[0,1].manager#USERDN";)
```

This ACI ensures that add permission is granted only to users whose bind DN matches the manager attribute of the parent entry.

Defining Access From a Specific IP Address

Using bind rules, you can indicate that the bind operation must originate from a specific IP address. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on an IP address is as follows:

`ip = "`*IP_address*`" or ip != "`*IP_address*`"`

The IP address must be expressed in dot notation.You can use the wildcard character (*) to include multiple machines. For example, the following string is valid:

```
ip = "12.123.1.*";
```

The bind rule is evaluated to be true if the client accessing the directory is located at the named IP address. This can be useful for allowing certain kinds of directory access only from a specific subnet or machine.

For example, you could use a wildcard IP address such as 12.3.45.* to specify a specific subnetwork or 123.45.6.*+255.255.255.115 to specify a subnetwork mask.

From the Server Console, you can define specific machines to which the ACI applies through the Access Control Editor. For more information, see “Creating ACIs From the Console,” on page 218.

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 host name is as follows:

`dns = "`*DNS_Hostname*`" or dns != "`*DNS_Hostname*`"`

CAUTION The `dns` keyword requires that the naming service used on your machine is DNS. If the name service is not DNS, you should use the `ip` keyword instead.

The `dns` keyword requires a fully qualified DNS domain name. Granting access to a host without specifying the domain creates a potential security threat. For example, the following expression is allowed but not recommended:

```
dns = "legend.eng";
```

You should use a fully qualified name such as:

```
dns = "legend.eng.siroe.com";
```

The dns keyword allows wildcards. For example:

```
dns = "*.siroe.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. Note that wildcards will not work if your system uses a naming service other than DNS. In such a case, if you want to restrict access to a particular domain, use the ip keyword, as described in "Defining Access From a Specific IP Address," on page 213.

Defining Access at a Specific Time of Day or Day of Week

You can use bind rules to specify that binding can only occur at a certain time of day or on a certain day of the week. For example, you can set a rule that will allow access only if it is between the hours of 8 am and 5 pm Monday through Friday. The time used to evaluate access rights is the time on the directory server, not the time on the client.

The LDIF syntax for setting a bind rule based on the time of day is as follows:

`timeofday` *operator* `"`*time*`"`

where *operator* can be one of the following symbols: equal to (=), not equal to (!=), greater than (>), greater than or equal to (>=), less than (<), or less than or equal to (<=).

The `timeofday` keyword requires a time of day expressed in hours and minutes in the 24 hour clock (0 to 2359).

NOTE	The time on the server is used for the evaluation, and not the time on the client.

The LDIF syntax for setting a bind rule based on the day in the week is as follows:

`dayofweek = "`*day1, day2 ...*`"`

The possible values for the dayofweek keyword are the English three-letter abbreviations for the days of the week: sun, mon, tue, wed, thu, fri, sat.

Examples

The following are examples of the `timeofday` and `dayofweek` syntax:

`timeofday = "1200";`

The bind rule is evaluated to be true if the client is accessing the directory at noon.

`timeofday != "0100";`

The bind rule is evaluated to be true if the client is accessing the directory at any time other than 1 am.

`timeofday > "0800";`

The bind rule is evaluated to be true if the client is accessing the directory at any time after 8 am.

`timeofday < "1800";`

The bind rule is evaluated to be true if the client is accessing the directory at any time before 6 pm.

`timeofday >= "0800";`

The bind rule is evaluated to be true if the client is accessing the directory at 8am or later.

`timeofday <= "1800";`

The bind rule is evaluated to be true if the client is accessing the directory at 6 pm or earlier.

`dayofweek = "Sun, Mon, Tue";`

The bind rule is evaluated to be true if the client is accessing the directory on Sunday, Monday, or Tuesday.

Defining Access Based on Authentication Method

You can set bind rules that state that a client must bind to the directory using a specific authentication method. The authentication methods available are:

- **None**

  Authentication is not required. This is the default. It represents anonymous access.

- **Simple**

  The client must provide a user name and password to bind to the directory.

- **SSL**

  The client must bind to the directory over a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection.

  In the case of SSL, the connection is established to the LDAPS second port; in the case of TLS, the connection is established through a Start TLS operation.In both cases, a certificate must be provided. For information on setting up SSL, see Chapter 11, “Managing SSL.”

- **SASL**

  The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. Note that iPlanet Directory Server does not provide a SASL module.

You cannot set up authentication-based bind rules through the Access Control Editor.

The LDIF syntax for setting a bind rule based on an authentication method is as follows:

`authmethod = "`*authentication_method*`"`

where *authentication_method* is **`none`**, **`simple`**, **`ssl`**, or **`"sasl`** *sasl_mechanism*`"`.

Examples

The following are examples of the `authmethod` keyword:

```
authmethod = "none";
```

Authentication is not checked during bind rule evaluation.

`authmethod = "simple";`

The bind rule is evaluated to be true if the client is accessing the directory using a username and password.

`authmethod = "ssl";`

The bind rule is evaluated to be true if the client authenticates to the directory using a certificate over LDAPS. This is not evaluated to be true if the client authenticates using simple authentication (bind DN and password) over ldaps.

`authmethod = "sasl DIGEST-MD5";`

The bind rule is evaluated to be true if the client is accessing the directory using the SASL DIGEST-MD5 mechanism. The other supported SASL mechanism is EXTERNAL.

Using Boolean Bind Rules

Bind rules can be complex expressions that use the Boolean expressions `AND`, `OR`, and `NOT` to set very precise access rules. You cannot use the Server Console to create Boolean bind rules. You must create an LDIF statement.

The LDIF syntax for a Boolean bind rule is as follows:

bind_rule [*boolean*][*bind_rule*][*boolean*][*bind_rule*]...;)

For example, the following bind rule will be evaluated to be true if the bind DN is a member of either the administrator's group or the mail administrator's group, and if the client is running from within the Siroe domain:

```
(groupdn = "ldap:///cn=administrators,dc=siroe,dc=com" or groupdn =
"ldap:///cn=mail administrators,dc=siroe,dc=com" and dns =
"*.siroe.com";)
```

The trailing semicolon (;) is a required delimiter that must appear after the final bind rule.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first
- All expressions from left to right
- NOT before AND or OR operators

The Boolean `OR` and Boolean `AND` operators have no order of precedence.

Consider the following Boolean bind rules:

`(`*bind_rule_A*`) OR (`*bind_rule_B*`)`

`(`*bind_rule_B*`) OR (`*bind_rule_A*`)`

Because Boolean expressions are evaluated from left to right, in the first case, bind rule A is evaluated before bind rule B, and in the second case, bind rule B is evaluated before bind rule A.

However, the Boolean `NOT` is evaluated *before* the Boolean `OR` and Boolean `AND`. Thus, in the following example:

`(`*bind_rule_A*`) AND NOT (`*bind_rule_B*`)`

bind rule B is evaluated before bind rule A despite the left-to-right rule.

Creating ACIs From the Console

You can use the Directory Server Console to view, create, edit, and delete access control instructions for your directory. This section provides general instructions for:

- Displaying the Access Control Editor
- Viewing Current ACIs
- Creating a New ACI
- Editing an ACI
- Deleting an ACI

See "Access Control Usage Examples," on page 224 for a collection of access control rules commonly used in Directory Server security policies, along with step-by-step instructions for using the Directory Server Console to create them.

The Access Control Editor does not enable you to construct some of the more complex ACIs when you are in Visual editing mode. In particular, from the Access Control Editor you cannot:

- Deny access (see "Permissions Syntax," on page 199)
- Create value-based ACIs (see "Targeting Attribute Values Using LDAP Filters," on page 194)
- Define parent access (see "Parent Access (parent Keyword)," on page 203)
- Create ACIs that contain Boolean bind rules (see "Using Boolean Bind Rules," on page 217)

- Generally, create ACIs that use the following keywords: `roledn`, `userattr`, authmethod

TIP In the Access Control Editor, you can click on the Edit Manually button at any time to check the LDIF representation of the changes you make through the graphical interface.

Displaying the Access Control Editor

1. Start the Directory Server Console. Log in using the bind DN and password of a privileged user such as the directory manager who has write access to the ACIs configured for the directory.

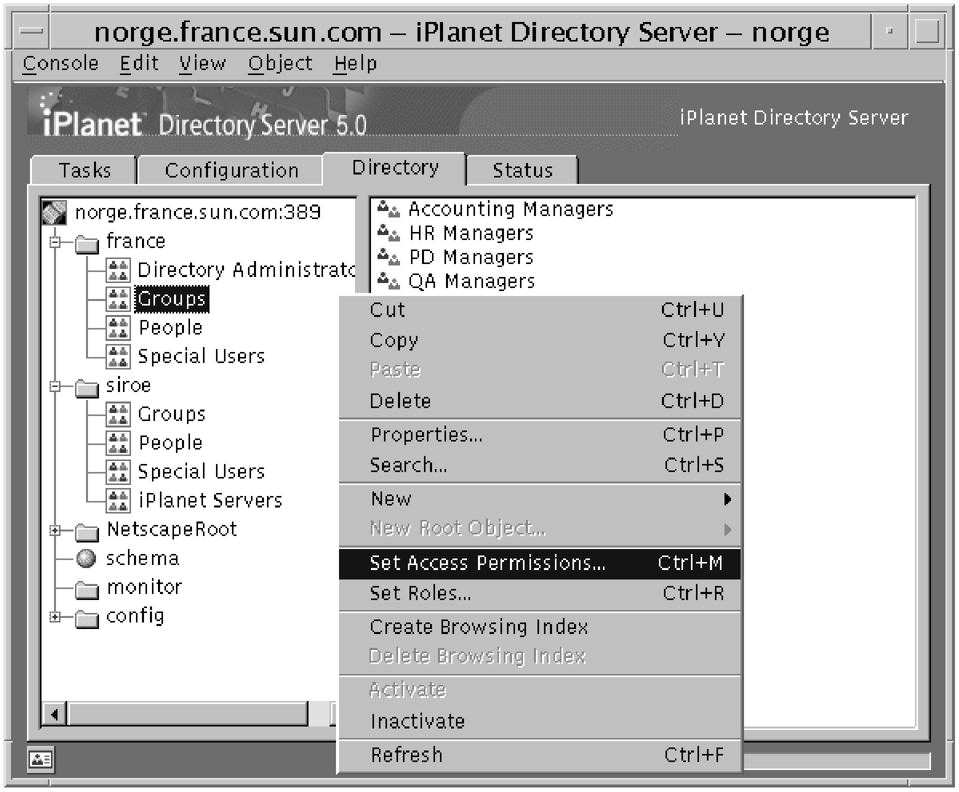
   For instructions, refer to "Using the Directory Server Console," on page 26.

2. On the Directory Server Console, select the Directory tab.

3. Right-click the entry in the navigation tree for which you want to set access control, and select Set Access Permissions from the pop-up menu (Figure 6-2 on page 220).

   Alternatively, highlight the entry, and select Set Access Permissions from the Object menu.

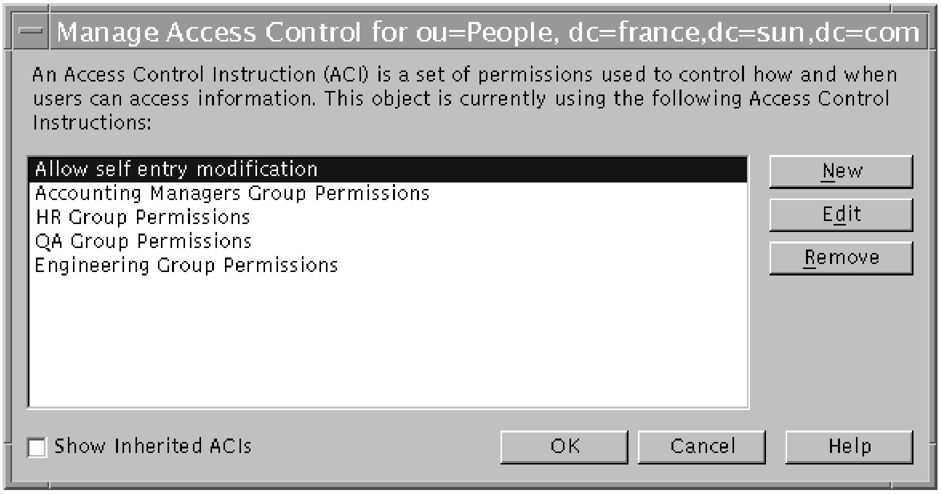
Figure 6-2 Selecting an Object in the Navigation Tree to Set Access Control



4. Click New.

   The Access Control Editor is displayed as shown in Figure 6-3 on page 221.

Figure 6-3 Access Control Editor Window



Manage Access Control for ou=People, dc=france,dc=sun,dc=com

An Access Control Instruction (ACI) is a set of permissions used to control how and when users can access information. This object is currently using the following Access Control Instructions:

Allow self entry modification
Accounting Managers Group Permissions
HR Group Permissions
QA Group Permissions
Engineering Group Permissions

For information on navigating through the Access Control dialog boxes, refer to the online help.

Viewing Current ACIs

If you want to see what ACIs apply to a particular subtree in your directory, follow these steps:

1. On the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. Check the checkbox for Show Inherited ACIs, if you want to display the full list of ACIs that apply to the entry.

Creating a New ACI

To create a new ACI:

1. Display the Access Control Editor.

   This task is explained in “Displaying the Access Control Editor,” on page 219.

   If the view displayed is different from Figure 6-3 on page 221, click the Edit Visually button.

2. Name the ACI, by typing a name in the ACI Name text box.

   The name can be any string you want to use to uniquely 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.

   In the Add Users and Groups window:

   a. Select a search area from the drop-down list, enter a search string in the Search field, and click the Search button.

      The search results are displayed in the list below.

   b. Highlight the entries you want in the search result list, and click the Add button to add them to the list of entries which have access permission.

   c. Click OK to dismiss the Add Users and Groups window.

      The entries you selected are now listed on the Users / Groups tab in the ACI editor.

4. In the Access Control Editor, click the Rights tab, and use the checkboxes to select the rights to grant.

5. Click the Targets tab, then click This Entry to display the node targeted by the ACI.

   You can change the value of the target DN, but the new DN must be a direct or indirect child of the selected entry.

   If you do not want every entry in the subtree under this node to be targeted by the ACI, you must enter a filter in the Filter for Sub-entries field.

   Additionally, you can restrict the scope of the ACI to only certain attributes by selecting the attributes you want to target in the attribute list.

6. Click the Hosts tab, then the Add button to display the Add Host Filter dialog box.

   You can specify a hostname or an IP address. If you specify an IP address, you can use the wildcard character (*).

7. Click the Times tab to display the table showing at what times access is allowed.

   By default, access is allowed at all times. You can change the access times by clicking and dragging the cursor over the table. You cannot select discrete blocks of time.

8. When you have finished editing the ACI, click OK.

   The ACI Editor is dismissed and the new ACI is listed in the ACI Manager window.

NOTE At any time during the creation of the ACI, you can click the Edit Manually button to display the LDIF statement that corresponds to your input. You can modify this statement, but your changes will not necessarily be visible in the graphical interface.

Editing an ACI

To edit an ACI:

1. On the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. In the Access Control Manager window, highlight the ACI that you want to edit, and click Edit.

   The Access Control Editor is displayed. For details on the information you can edit using this dialog box, refer to the online help.

3. Make the changes you want under the various tabs of the Access Control Editor.

4. When you have finished editing the ACI, click OK.

   The ACI Editor is dismissed, and the modified ACI is listed in the ACI Manager.

Deleting an ACI

To delete an ACI:

1. On the Directory tab, right-click the top entry in the subtree, and choose Set Access Permissions from the pop-up menu.

   The Access Control Manager window is displayed. It contains the list of ACIs belonging to the entry.

2. In the Access Control Manager window, select the ACI that you want to delete.

3. Click Remove.

   The ACI is no longer listed in the Access Control Manager.

Access Control Usage Examples

The examples provided in this section illustrate how an imaginary ISP company, Siroe, would implement its access control policy. All the examples explain how to perform a given task from the Console and using an LDIF file.

Siroe's business is to offer a web hosting service and internet access. Part of Siroe's web hosting service is to host the directories of client companies. Siroe actually hosts and partially manages the directories of two medium-sized companies, Company333, and Company999. It also provides internet access to a number of individual subscribers.

These are the access control rules that Siroe wants to put in place:

- Grant anonymous access for read, search, and compare to the entire Siroe tree for Siroe employees (see "Granting Anonymous Access," on page 225).
- Grant write access to Siroe employees for personal information for such as `homeTelephoneNumber`, `homeAddress` (see "Granting Write Access to Personal Entries," on page 227).
- Grant Siroe employees the right to add any role to their entry, except certain critical roles (see "Restricting Access to Key Roles," on page 231).
- Grant the Siroe Human Resources group all rights on the entries in the People branch (see "Granting a Group Full Access to a Suffix," on page 232).
- Grant all Siroe employees the right to create group entries under the Social Committee branch of the directory, and to delete group entries that they own (see "Granting Rights to Add and Delete Group Entries," on page 233).

- Grant all Siroe employees the right to add themselves to group entries under the Social Committee branch of the directory (see “Allowing Users to Add or Remove Themselves From a Group,” on page 240).
- Grant access to the directory administrator (role) of Company333 and Company999 on their respective branches of the directory tree, with certain conditions such as SSL authentication, time and date restrictions, and specified location (see “Granting Conditional Access to a Group or Role,” on page 235).
- Grant individual subscribers access to their own entries (see “Granting Write Access to Personal Entries,” on page 227).
- Deny individual subscribers access to the billing information in their own entries (see “Denying Access,” on page 238).
- 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 slave server outside of the firewall and updated once a day.) See “Granting Anonymous Access,” on page 225 and “Setting a Target Using Filtering,” on page 240.

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 Siroe internally, and is illustrated in the ACI “Anonymous Siroe” example.

As an ISP, Siroe also wants to advertise the contact information of all of its subscribers by creating a public phonebook accessible to the world. This is illustrated in the ACI “Anonymous World” example.

ACI “Anonymous Siroe”

In LDIF, to grant read, search, and compare permissions to the entire Siroe tree to Siroe employees, you would write the following statement:

```
aci: (targetattr !="userPassword")(version 3.0; acl "Anonymous
Siroe"; allow (read, search, compare) userdn= "ldap:///anyone" and
dns="*.siroe.com";)
```

This example assumes that the `aci` is added to the `dc=siroe,dc=com entry`. Note that the userPassword attribute is excluded from the scope of the ACI.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the siroe 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. On the Users/Groups tab, in the ACI name field, type "Anonymous Siroe". Check that All Users is displayed in the list of users granted access permission.
4. On the Rights tab, tick the checkboxes for read, compare, and search rights. Make sure the other checkboxes are clear.
5. On the Targets tab, click This Entry to display the `dc=siroe,dc=com` suffix in the target directory entry field. In the attribute table, locate the `userPassword` attribute and clear the corresponding checkbox.

   All other checkboxes should be ticked. This task is made easier if you click the Name header to organize the list of attributes alphabetically.
6. On the Hosts tab, click Add, and in the DNS host filter field, type `*.siroe.com`. Click OK to dismiss the dialog box.
7. Click OK in the Access Control Editor window.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI “Anonymous World”

In LDIF, to grant read and search access of the individual subscribers subtree to the world, while denying access to information on unlisted subscribers, you could write the following statement:

```
aci: (targetfilter= "(!(unlistedSubscriber=yes))")
(targetattr="homePostalAddress || homePhone || mail") (version 3.0;
acl "Anonymous World"; allow (read, search) userdn=
"ldap:///anyone";)
```

This example assumes that the ACI is added to the `ou=subscribers,dc=siroe, dc=com` entry. It also assumes that every subscriber entry has an `unlistedSubscriber` attribute which is set to yes or no. The target definition filters out the unlisted subscribers based on the value of this attribute. For details on the filter definition, refer to “Setting a Target Using Filtering,” on page 240.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the Subscribers entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Anonymous World". Check that All Users is displayed in the list of users granted access permission.
4. On the Rights tab, tick the checkboxes for read, and search rights. Make sure the other checkboxes are clear.
5. On the Targets tab, click This Entry to display the `dc=subscribers, dc=siroe,dc=com` suffix in the target directory entry field.
    a. In the filter for subentries field, type the following filter:

       `(!(unlistedSubscriber=yes))`

    b. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

       All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.
6. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Granting Write Access to Personal Entries

Many directory administrators want to allow internal users to change some but not all of the attributes in their own entry. The directory administrators at Siroe want to allow users to change their own password, home telephone number, and home address, but nothing else. This is illustrated in the ACI “Write Siroe” example.

It is also Siroe’s policy to let their subscribers update their own personal information in the Siroe tree provided that they establish an SSL connection to the directory. This is illustrated in the ACI “Write Subscribers” example.

ACI “Write Siroe”

NOTE	By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant Siroe employees the right to update their password, home telephone number and home address, you would write the following statement:

```
aci: (targetattr="userPassword || homePhone || homePostalAddress")
(version 3.0; acl "Write Siroe"; allow (write) userdn=
"ldap:///self" and dns="*.siroe.com";)
```

This example assumes that the ACI is added to the `ou=siroe-people,dc=siroe, dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the siroe 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. On the Users/Groups tab, in the ACI name field, type "Write Siroe". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, tick the checkbox for write right. Make sure the other checkboxes are clear.

5. On the Targets tab, click This Entry to display the `dc=siroe,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `userPassword` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then clikc the Name header to organize them alphabetically, and select the appropriate ones.

6. On the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.siroe.com`. Click OK to dismiss the dialog box.

7. Click OK in the Access Control Editor window.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI "Write Subscribers"

NOTE By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant Siroe subscribers the right to update their password and home telephone number, you would write the following statement:

```
aci: (targetattr="userPassword || homePhone") (version 3.0; acl
"Write Subscribers"; allow (write) userdn= "ldap://self" and
authmethod="ssl";)
```

This example assumes that the `aci` is added to the `ou=subscribers,dc=siroe, dc=com` entry.

Note that Siroe subscribers do not have write access to their home address, because they might delete the attribute, and Siroe needs that information for billing. Therefore, the home address is business-critical information.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the Subscribers entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Write Subscribers". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, tick the checkbox for write. Make sure the other checkboxes are clear.
5. On the Targets tab, click This Entry to display the `dc=subscribers, dc=siroe,dc=com` suffix in the target directory entry field.
   a. In the filter for subentries field, type the following filter:

      ```
      (!(unlistedSubscriber=yes))
      ```
   b. In the attribute table, tick the checkboxes for the `homePhone`, `homePostalAddress`, and `mail` attributes.

      All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then click the Name header to organize them alphabetically, and select the appropriate ones.
6. If you want users to authenticate using SSL, switch to manual editing by clicking the Edit Manually button and add `authmethod=ssl` to the LDIF statement so that it reads as follows:

   ```
   (targetattr="homePostalAddress || homePhone || mail") (version
   3.0; acl "Write Subscribers"; allow (write) (userdn=
   "ldap:///self") and authmethod="ssl";)
   ```
7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Restricting Access to Key Roles

You can use role definitions in the directory to identify functions that are critical to your business, the administration of your network and directory, or another purpose.

For example, you might create a `superAdmin` role by identifying a subset of your system administrators that are available at a particular time of day and day of the week at corporate sites worldwide. Or you might want to create a `First Aid` role that includes all members of staff on a particular site that have done first aid training. For information on creating role definitions, refer to “Using Roles,” on page 156.

When a role gives any sort of privileged user rights over critical corporate or business functions, you should consider restricting access to that role. For example, at Siroe, employees can add any role to their own entry, except the `superAdmin` role. This is illustrated in the ACI “Roles” example.

ACI “Roles”

In LDIF, to grant Siroe employees the right to add any role to their own entry, except the `superAdmin` role, you would write the following statement:

```
aci: (targattrfilters="add=nsRoleDN:(nsRoleDN !=
"cn=superAdmin,dc=siroe,dc=com")") (version 3.0; acl "Roles";
allow (write) userdn= "ldap:///self" and dns="*.siroe.com";)
```

This example assumes that the ACI is added to the `ou=siroe-people,dc=siroe, dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the siroe 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. On the Users/Groups tab, in the ACI name field, type "Roles". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.

d. Click OK to dismiss the Add Users and Groups dialog box.

4. On the Rights tab, tick the checkbox for write. Make sure the other checkboxes are clear.

5. On the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.siroe.com`. Click OK to dismiss the dialog box.

6. To create the value-based filter for roles, switch to manual editing by clicking the Edit Manually button. Add the following to the beginning of the LDIF statement:

   ```
   (targattrfilters="add=nsRoleDN:(nsRoleDN != "cn=superAdmin,
   dc=siroe,dc=com")")
   ```

   The LDIF statement should read as follows:

   ```
   (targattrfilters="add=nsRoleDN:(nsRoleDN != "cn=superAdmin,
   dc=siroe,dc=com")") (targetattr = “*”) (target =
   "ldap:///dc=siroe,dc=com") (version 3.0; acl "Roles"; allow
   (write) (userdn = "ldap:///self") and (dns="*.siroe.com");)
   ```

7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Granting a Group Full Access to a Suffix

Most directories have a group that is used to identify certain corporate functions. These groups can be given full access to all or part of the directory. By applying the access rights to the group, you can avoid setting the access rights for each member individually. Instead, you grant users these access rights simply by adding them to the group.

For example, when you install the Directory Server using the Typical Install process, an Administrators group with full access to the directory is created by default.

At Siroe, the Human Resources group is allowed full access to the `ou=siroe-people` branch of the directory so that they can update the employee database. This is illustrated in the ACI “HR” example.

ACI “HR”

In LDIF, to grant the HR group all rights on the employee branch of the directory, you would use the following statement:

```
aci: (version 3.0; acl "HR"; allow (all) userdn=
"ldap:///cn=HRgroup,ou=siroe-people,dc=siroe,dc=com";)
```

This example assumes that the ACI is added to the `ou=siroe-people,dc=siroe, dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the siroe-people entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "HR". In the list of users granted access permission, do the following:
    a. Select and remove All Users, then click Add.

       The Add Users and Groups dialog box is displayed.
    b. Set the Search area to Users and Groups, and type “HRgroup” in the Search for field.

       This example assumes that you have created an HR group or role. For more information on groups and roles, see Chapter 5, “Advanced Entry Management.”
    c. Click the Add button to list the HR group in the list of users who are granted access permission.
    d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, click the Check All button.

   All checkboxes are ticked, except for Proxy rights.
5. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Granting Rights to Add and Delete Group Entries

Some organizations want to allow employees to create entries in the tree if it can increase their efficiency, or if it can contribute to the corporate dynamics.

At Siroe for example, there is an active social committee that is organized into various clubs: tennis, swimming, skiing, role-playing, etc. Any Siroe employee can create a group entry representing a new club. This is illustrated in the ACI “Create Group” example. Any Siroe employee can become a member of one of these groups. This is illustrated in ACI “Group Members” under “Allowing Users to Add or Remove Themselves From a Group,” on page 240. Only the group owner can modify or delete a group entry. This is illustrated in the ACI “Delete Group” example.

ACI “Create Group”

In LDIF, to grant Siroe employees the right to create a group entry under the `ou=Social Committee` branch, you would write the following statement:

```
aci: (target="ldap:///ou=social committee,dc=siroe,dc=com)
(targattrfilters="add=objectClass:(objectClass=groupOfNames)")
(version 3.0; acl "Create Group"; allow (read,search,add)
(userdn= "ldap:///uid=*,ou=siroe-people,dc=siroe,dc=com") and
dns="*.siroe.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=siroe,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the Social Committee entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Create Group". In the list of users granted access permission, do the following:
    a. Select and remove All Users, then click Add.

       The Add Users and Groups dialog box is displayed.
    b. Set the Search area to Special Rights, and select All Authenticated Users from the Search results list.
    c. Click the Add button to list All Authenticated Users in the list of users who are granted access permission.
    d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, tick the checkbox for read, search, and add. Make sure the other checkboxes are clear.

5. On the Targets tab, click This Entry to display the `ou=social committee, dc=siroe,dc=com` suffix in the target directory entry field.
6. On the Hosts tab, click Add to display the Add Host Filter dialog box. In the DNS host filter field, type `*.siroe.com`. Click OK to dismiss the dialog box.
7. To create the value-based filter that will allow employees to add only group entries to this subtree, switch to manual editing by clicking the Edit Manually button. Add the following to the beginning of the LDIF statement:

   ```
   (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
   ```

   The LDIF statement should read as follows:

   ```
   (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
   (targetattr = "*") (target="ldap:///ou=social
   committee,dc=siroe,dc=com) (version 3.0; acl "Create Group";
   allow (read,search,add) (userdn= "ldap:///all") and
   (dns="*.siroe.com"); )
   ```

8. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI “Delete Group”

In LDIF, to grant Siroe employees the right to modify or delete a group entry which they own under the ou=Social Comittee branch, you would write the following statement:

```
aci: (target="ou=social committee,dc=siroe,dc=com)
(targattrfilters="del=objectClass:(objectClass=groupOfNames)")
(version 3.0; acl "Delete Group"; allow (write,delete) userattr=
"owner#GROUPDN";)
```

This example assumes that the `aci` is added to the `ou=social committee, dc=siroe,dc=com` entry.

Using the Console is not an effective way of creating this ACI because you would have to use manual editing mode to create the target filter, and to check group ownership.

Granting Conditional Access to a Group or Role

In many cases, when you grant a group or role privileged access to the directory, you want to ensure that those privileges are protected from intruders trying to impersonate your privileged users. Therefore, in many cases, access control rules that grant critical access to a group or role are often associated with a number of conditions.

Siroe, for example, has created a Directory Administrator role for each of its hosted companies, Company333 and Company999. 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, Company333 and Company999 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 am and 6 pm, Monday through Thursday, and
- Access requested from a specified IP address for each company.

These conditions are illustrated in a single ACI for each company, ACI “Company333” and ACI “Company999”. Because the content of these ACIs is the same, the examples below illustrate the “Company333 “ ACI only.

ACI “Company333”

In LDIF, to grant Company333 full access to their own branch of the directory under the conditions stated above, you would write the following statement:

```
aci: (target="ou=Company333,ou=corporate-clients,dc=siroe,dc=com")
(version 3.0; acl "Company333"; allow (all) (roledn=
"ldap:///cn=DirectoryAdmin,ou=Company333,ou=corporate-clients,
dc=siroe,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=Company333, ou=corporate-clients,dc=siroe,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the Company333 entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Company333". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area to Users and Groups, and type DirectoryAdmin in the Search For field.

      This example assumes that you have created an administrators role with a `cn` of `DirectoryAdmin`.

   c. Click the Add button to list the administrators role in the list of users who are granted access permission.

   d. Click OK to dismiss the Add Users and Groups dialog box.

4. On the Rights tab, click the Check All button.

5. On the Targets tab, click This Entry to display the `ou=Company333,ou=corporate-clients,dc=siroe,dc=com` suffix in the target directory entry field.

6. On the Hosts tab, click Add to display the Add Host Filter dialog box. In the IP address host filter field, type `255.255.123.234`. Click OK to dismiss the dialog box.

   The IP address must be a valid IP address for the host machine that the Company333 administrators will use to connect to the Siroe directory.

7. On the Times tab, select the block time corresponding to Monday through Thursday, and 8 am to 6 pm.

   A message appears below the table that specifies what time block you have selected.

8. To enforce SSL authentication from Company333 administrators, switch to manual editing by clicking the Edit Manually button. Add the following to the end of the LDIF statement:

   ```
   and (authmethod="ssl")
   ```

   The LDIF statement should be similar to:

   ```
   aci: (targetattr = "*")
   (target="ou=Company333,ou=corporate-clients,dc=siroe,dc=com")
   (version 3.0; acl "Company333"; allow (all) (roledn=
   "ldap:///cn=DirectoryAdmin,ou=Company333,ou=corporate-clients,
   dc=siroe,dc=com") and (dayofweek="Mon,Tues,Wed,Thu") and
   (timeofday >= "0800" and timeofday <= "1800") and
   (ip="255.255.123.234") and (authmethod="ssl"); )
   ```

9. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Denying Access

If your directory holds business-critical information, you might specifically want to deny access to it.

For example, Siroe wants all subscribers to be able to read billing information such as connection time or account balance under their own entries, but explicitly wants to deny write access to that information. This is illustrated in ACI "Billing Info Read" and ACI "Billing Info Deny" respectively.

ACI "Billing Info Read"

In LDIF, to grant subscribers permission to read billing information in their own entry, you would write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version 3.0;
acl "Billing Info Read"; allow (search,read) userdn=
"ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema, and that the ACI is added to the `ou=subscribers,dc=siroe,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the subscribers entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Billing Info Read". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, tick the checkboxes for search and read rights. Make sure the other checkboxes are clear.

5. On the Targets tab, click This Entry to display the `ou=subscribers, dc=siroe,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `connectionTime` and `accountBalance` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then clikc the Name header to organize them alphabetically, and select the appropriate ones.

   This example assumes that you have added the the `connectionTime` and `accountBalance` attributes to the schema.

6. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

ACI "Billing Info Deny"

In LDIF, to deny subscribers permission to modify billing information in their own entry, you would write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version 3.0;
acl "Billing Info Deny"; deny (write) userdn= "ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema, and that the ACI is added to the `ou=subscribers,dc=siroe,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the subscribers entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Billing Info Deny". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.

   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select Self from the Search results list.
   c. Click the Add button to list Self in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.

4. On the Rights tab, tick the checkbox for write. Make sure the other checkboxes are clear.

5. Click the Edit Manually button and in the LDIF statement that is displayed, change the word **`allow`** to **`deny`**.

6. On the Targets tab, click This Entry to display the `ou=subscribers, dc=siroe,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkboxes for the `connectionTime` and `accountBalance` attributes.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then clikc the Name header to organize them alphabetically, and select the appropriate ones.

   This example assumes that you have added the the `connectionTime` and `accountBalance` attributes to the schema.

7. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Setting a Target Using Filtering

If you want to set access controls that allow access to a number of entries that are spread across the directory, you may want to use a filter to set the target. Keep in mind that because search filters do not directly name the object for which you are managing access, it is easy to unintentionally allow or deny access to the wrong objects, especially as your directory becomes more complex. Additionally, filters can make it difficult for you to troubleshoot access control problems within your directory.

The following procedure shows you how to grant user `bjensen` write access to the department number, home phone number, home postal address, JPEG photo, and manager attributes for all members of the accounting organization.

Before you can set these permissions, you must create the accounting branch point (`ou=accounting,dc=siroe,dc=com`). You can create organizational unit branch points using the directory tab on the Directory Server Console.

Allowing Users to Add or Remove Themselves From a Group

Many directories set ACIs that allow users to add or remove themselves from groups. This is useful, for example, for allowing users to add and remove themselves from mailing lists.

At Siroe, employees can add themselves to any group entry under the `ou=social committee` subtree. This is illustrated in the ACI “Group Members” example.

ACI “Group Members”

In LDIF, to grant Siroe employees the right to add or delete themselves from a group, you would write the following statement:

```
aci: (targettattr="member")(version 3.0; acl "Group Members";
allow (selfwrite)
(userdn= "ldap:///uid=*,ou=siroe-people,dc=siroe,dc=com") ;)
```

This example assumes that the ACI is added to the `ou=social committee, dc=siroe,dc=com` entry.

From the Console, you can set this permission by doing the following:

1. On the Directory tab, right click the siroe-people entry under the siroe 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. On the Users/Groups tab, in the ACI name field, type "Group Members". In the list of users granted access permission, do the following:
   a. Select and remove All Users, then click Add.

      The Add Users and Groups dialog box is displayed.
   b. Set the Search area in the Add Users and Groups dialog box to to Special Rights, and select All Authenticated Users from the Search results list.
   c. Click the Add button to list All Authenticated Users in the list of users who are granted access permission.
   d. Click OK to dismiss the Add Users and Groups dialog box.
4. On the Rights tab, tick the checkbox for selfwrite. Make sure the other checkboxes are clear.
5. On the Targets tab, type `dc=siroe,dc=com` suffix in the target directory entry field. In the attribute table, tick the checkbox for the `member` attribute.

   All other checkboxes should be clear. This task is made easier if you click the Check None button to clear the checkoxes for all attributes in the table, then clikc the Name header to organize them alphabetically, and select the appropriate ones.

6. Click OK.

   The new ACI is added to the ones listed in the Access Control Manager window.

Defining Permissions for DNs That Contain a Comma

DNs that contain commas require special treatement within your LDIF ACI statements. In the target and bind rule portions of the ACI statement, commas must be escaped by a single backslash (\). The following example illustrates this syntax:

```
dn: dc=Siroe Bolivia\, S.A.,dc=com
objectClass: top
objectClass: organization
aci: (target="ldap:///dc=Siroe Bolivia\, S.A.,dc=com")(targetattr=*)
(version 3.0; acl "aci 2"; allow (all)
groupdn = "ldap:///cn=Directory Administrators,dc=Siroe Bolivia\,
S.A.,dc=com";)
```

Proxied Authorization ACI Example

For this example, suppose:

- The client application's bind DN is `"uid=MoneyWizAcctSoftware, ou=Applications,dc=siroe,dc=com"`.
- The targeted subtree to which the client application is requesting access is `ou=Accounting,dc=siroe,dc=com`.
- An Accounting Administrator with access permissions to the `ou=Accounting,dc=siroe,dc=com` subtree exists in the directory.

In order for the client application to gain access to the Accounting subtree (using the same access permissions as the Accounting Administrator):

- The Accounting Administrator must have access permissions to the `ou=Accounting,dc=siroe,dc=com` subtree. For example, the following ACI grants all rights to the Accounting Administrator entry:

```
aci: (target="ldap:///ou=Accounting,dc=siroe,dc=com")
(targetattr="*") (version 3.0; acl "allowAll-AcctAdmin"; allow (all)
 userdn="uid=AcctAdministrator,ou=Administrators,dc=siroe,dc=com")
```

- The following ACI granting proxy rights to the client application must exist in the directory:

```
aci: (target="ldap:///ou=Accounting,dc=siroe,dc=com")
(targetattr="*") (version 3.0; acl "allowproxy-accountingsoftware";
allow (proxy)
userdn="uid=MoneyWizAcctSoftware,ou=Applications,dc=siroe,dc=com")
```

With this ACI in place, the MoneyWizAcctSoftware client application can bind to the directory and send an LDAP command such as `ldapsearch` or `ldapmodify` that requires the access rights of the proxy DN.

In the above example, if the client wanted to perform an `ldapsearch` command, the command would include the following controls:

```
#ldapmodify -D "uid=MoneyWizAcctSoftware,
ou=Applications,dc=siroe,dc=com" -w secretpwd
-y "uid=AcctAdministrator,ou=Administrators,dc=siroe,dc=com"
```

Note that the client binds as itself, but is granted the privileges of the proxy entry. The client does not need the password of the proxy entry.

NOTE You cannot use the directory manager's DN (Root DN) as a proxy DN. In addition, if Directory Server receives more than one proxied authentication control, an error is returned to the client application and the bind attempt is unsuccessful.

Viewing the ACIs for an Entry

You can view all the ACIs under a single suffix in the directory by running the following `ldapsearch` command:

`ldapsearch -h` *host* `-p` *port* `-b` *baseDN* `-D` *rootDN* `-w` *rootPassword* `(aci=*) aci`

See *iPlanet Directory Server Configuration, Command, and File Reference* for information on using the `ldapsearch` utility.

From the Console, you can view all of the ACIs that apply to a particular entry through the Access Control Manager.

1. In the Directory Console, on the Directory tab, right-click the entry in the navigation tree, and select Set Access Permissions.

   The Access Control Manager is displayed. It contains a list of the ACIs belonging to the selected entry.

2. Check the Show Inherited ACIs checkbox to display all ACIs created on entries above the selected entry that also apply.

Advanced Access Control: Using Macro ACIs

In organizations that use repeating directory tree structures, it is possible to optimize the number of ACIs used in the directory by using macros. Reducing the number of ACIs in your directory tree makes it easier to manage your access control policy, and improves the efficiency of ACI memory usage.

Macros are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use a macro to represent a DN in the target portion of the ACI, or in the bind rule portion, or both. In practice, when Directory Server gets an incoming LDAP operation, the ACI macros are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource. Directory Server then evaluates the ACI normally.

Macro ACI Example

The benefits of macro ACIs and how they work are best explained using an example. Figure 6-4 on page 245 shows a directory tree in which using macro ACIs is an effective way of reducing the overall number of ACIs.

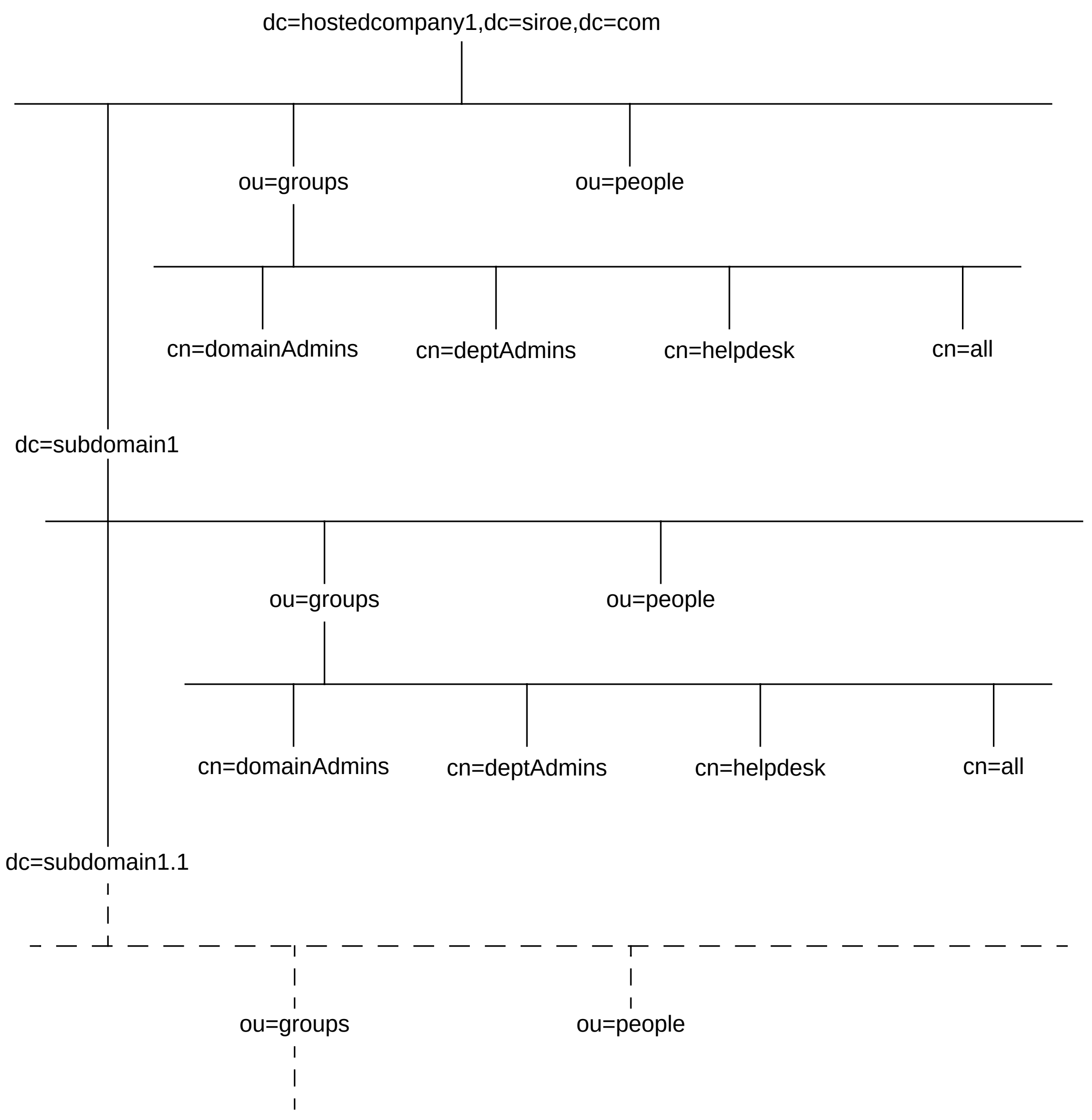
In this illustration, note the repeating pattern of subdomains with the same tree structure (ou=groups, ou=people). This pattern is also repeated across the tree, because the Siroe directory tree stores the following suffixes `dc=hostedCompany2,dc=siroe,dc=com`, and `dc=hostedCompany3,dc=siroe,dc=com`.

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=siroe,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=siroe,
dc=com";)
```

This ACI grants read and search rights to the DomainAdmins group to any entry in the `dc=hostedCompany1,dc=siroe,dc=com` tree.

Figure 6-4 Example directory tree for Macro ACIs



The following ACI is located on the `dc=hostedCompany1,dc=siroe,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=siroe,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany1,`
`dc=siroe,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=siroe,dc=com";)
```

The following ACI is located on the `dc=hostedCompany2,dc=siroe,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=siroe,dc=com";)
```

The following ACI is located on the `dc=subdomain1,dc=hostedCompany2,`
`dc=siroe,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=siroe,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=siroe,dc=com` node. This ACI reads as follows:

```
aci: (target=”ldap:///ou=Groups,($dn),dc=siroe,dc=com”)
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=siroe,dc=com";)
```

Note that the target keyword which was not previously used needs to be introduced.

In the example above, the number of ACIs is reduced from four to one. However, the real benefit is a factor of how many repeating patterns you have down and across your directory tree.

Macro ACI Syntax

Macro ACIs include the following types of expressions to replace a DN or part of a DN:

- ($dn)
- [$dn]
- ($attr.*attrName*), where *attrName* represents an attribute contained in the target entry

To simplify the discussion in this section, the ACI keywords used to provide bind credentials such as `userdn`, `roledn`, `groupdn`, and `userattr`, are collectively called the *subject*, as opposed to the target of the ACI. Macro ACIs can be used in the target part or the subject part of an ACI.

Table 6-3 shows in what parts of the ACI you can use DN macros:

Table 6-3 Macros in ACI Keywords

Macro	**ACI Keyword**
($dn)	target, targetfilter, userdn, roledn,groupdn, userattr
[$dn]	targetfilter, userdn, roledn, groupdn, userattr
($attr.*attrName*)	userdn, roledn, groupdn, userattr

The following restrictions apply:

- If you use ($dn) in `targetfilter`, `userdn`, `roledn`,`groupdn`, `userattr`, you *must* define a target that contains ($dn).
- If you use [$dn] in `targetfilter`, `userdn`, `roledn`,`groupdn`, `userattr`, you *must* define a target that contains ($dn).

In short, you when using any macro, you *always* need a target definition that contains the ($dn) macro.

You can combine the ($dn) macro and the ($attr.*attrName*) macro.

Macro Matching for ($dn)

The ($dn) macro is replaced by the matching part of the resource targeted in an LDAP request. For example, you have an LDAP request targeted at the `cn=all, ou=groups,dc=subdomain1,dc=hostedCompany1,dc=siroe,dc=com` entry, and an ACI that defines the target as follows:

```
(target="ldap:///ou=Groups,($dn),dc=siroe,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=*,($dn),dc=siroe,dc=com")
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,($dn),dc=siroe,dc=com";)
```

In this case, if the string matching ($dn) in the target is `dc=subdomain1, dc=hostedCompany1`, then the same string is used in the subject. The ACI above is expanded as follows:

```
aci: (target="ldap:///ou=Groups,dc=subdomain1,dc=hostedCompany1,
dc=siroe,dc=com") (version 3.0; acl "Domain access"; allow
(read,search) groupdn="ldap:///cn=DomainAdmins,ou=Groups,
dc=subdomain1,dc=hostedCompany1,dc=siroe,dc=com";)
```

Once the macro has been expanded, Directory Server evaluates the ACI following the normal process to determine whether access is granted or not.

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=siroe,dc=com` subtree, and the following ACI:

```
aci: (target="ldap:///ou=Groups,($dn),dc=siroe,dc=com")
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=siroe,dc=com";)
```

The steps for expanding this ACI are as follows:

1. ($dn) in target matches `dc=subdomain1,dc=hostedCompany1`.

2. Replace [$dn] in subject with `dc=subdomain1,dc=hostedCompany1`.

   The result is `groupdn="ldap:///cn=DomainAdmins,ou=Groups, dc=subdomain1,dc=hostedCompany1,dc=siroe, dc=com"`. If the bind DN is a member of that group, the matching process stops, and the ACI is evaluated. If it does not match, the process continues.

3. Replace [$dn] in subject with `dc=hostedCompany1`.

   The result is `groupdn="ldap:///cn=DomainAdmins,ou=Groups, dc=hostedCompany1,dc=siroe,dc=com"`. In this case, if the bind DN is not a member of that group, the ACI is not evaluated. If it is a member, the ACI is evaluated.

The advantage of the [$dn] macro is that it provides a flexible way of granting access to domain-level administrators to *all* the subdomains in the directory tree. Therefore, it is useful for expressing a hierarchical relationship between domains.

For example, consider the following ACI:

```
aci: (target="ldap:///ou=*, ($dn),dc=siroe,dc=com")
(targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
(version 3.0; acl "Domain access"; allow (read,search)
groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=siroe,dc=com";)
```

It grants access to the members of `cn=DomainAdmins,ou=Groups, dc=hostedCompany1,dc=siroe,dc=com` to all of the subdomains under `dc=hostedCompany1`, so an administrator belonging to that group could access, for example, the subtree `ou=people, dc=subdomain1.1, dc=subdomain1`.

However, at the same time, members of `cn=DomainAdmins,ou=Groups, dc=subdomain1.1`, would be denied access to the `ou=people,dc=hostedCompany1` and `ou=people,dc=hostedCompany1` nodes.

Macro Matching for ($attr.*attrName*)

The ($attr.*attrname*) macro is always used in the subject part of a DN. For example, you could define the following `roledn`:

```
roledn = "ldap:///cn=DomainAdmins,($attr.ou)"
```

Now, assume the server receives an LDAP operation targeted at the following entry:

```
dn: cn=Heather Blue, ou=People, dc=HostedCompany1, dc=siroe, dc=com
cn: Heather Blue
sn: Blue
ou: Engineering, dc=HostedCompany1, dc=siroe, dc=com
...
```

In order to evaluate the `roledn` part of the ACI, the server looks at the ou attribute stored in the targeted entry, and uses the value of this attribute to expand the macro. Therefore, in the example, the `roledn` is expanded as follows:

```
roledn = "ldap:///cn=DomainAdmins,ou=Engineering,dc=HostedCompany1,
dc=siroe,dc=com"
```

The Directory Server then evaluates the ACI according to the normal ACI evaluation algorithm.

When an attribute is multi-valued, each value is used to expand the macro, and the first one that provides a successful match is used.

Consider this example:

```
dn: cn=Heather Blue,ou=People,dc=HostedCompany1,dc=siroe,dc=com
cn: Heather Blue
sn: Blue
ou: Engineering, dc=HostedCompany1, dc=siroe, dc=com
ou: People, dc=HostedCompany1,dc=siroe, 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=siroe,dc=com"
```

```
roledn = "ldap:///cn=DomainAdmins,ou=People,dc=HostedCompany1,
dc=siroe,dc=com"
```

Access Control and Replication

ACIs are stored as attributes of entries, therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated like any other attribute.

ACIs are always evaluated on the Directory Server that services the incoming LDAP requests. This means that when a consumer server receives an update request, it will return a referral to the master server before evaluating whether the request can be serviced or not on the master.

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. On the Console, click the Directory tab, right click the config node, and choose Properties from the pop-up menu.

   This displays the Property Editor for the `cn=config` entry.

2. Scroll down the list of attribute value pairs to locate the `nsslapd-errorlog-level` attribute.

3. Add 128 to the value already displayed in the `nsslapd-errorlog-level` value field.

   For example, if the value already displayed is 8192 (replication debugging), you should change the value to 8320. For complete information on error log levels, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

4. Click OK to dismiss the Property Editor.

Compatibility with Earlier Releases

Some ACI keywords that were used in earlier releases of Directory Server have been deprecated in iPlanet Directory Server 5.0. However, for reasons of backward compatibility, they are still supported. These keywords are:

- `userdnattr`
- `groupdnattr`

Therefore, if you have set up a replication agreement between a legacy supplier server and a consumer Directory Server 5.0, you should not encounter any problems in the replication of ACIs.

Chapter 7

User Account Management

When a user connects to your directory server, first the user is authenticated. Then, the directory can grant access rights and resource limits to the user depending upon the identity established during authentication.

This chapter describes tasks for user account management, including configuring the password and account lockout policy for your directory, denying groups of users access to the directory, and limiting system resources available to users depending upon their bind DNs.

This chapter contains the following sections:

- Managing the Password Policy
- Inactivating Users and Roles
- Setting Resource Limits Based on the Bind DN

Managing the Password Policy

A password policy minimizes the risks of using passwords by enforcing the following:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords

Once you have established a password policy for your directory, you can protect your user passwords from potential threads by configuring an account lockout policy. Account lockout protects against hackers who try to break into the directory by repeatedly guessing a user's password.

This section provides information about configuring your password and account lockout policies. It includes the following procedures:

- "Configuring the Password Policy," on page 254
- "Setting User Passwords," on page 259
- "Configuring the Account Lockout Policy," on page 260
- "Managing the Password Policy in a Replicated Environment," on page 262

Configuring the Password Policy

The password policy you configure applies to all users within the directory except for the Directory Manager. Your password policy is comprised of the following information:

Password add and modify information. The password information includes password syntax and password history details.

Bind information. The bind information includes tracking bind failures and password aging attributes.

This section describes the following procedures for configuring your password policy:

- "Configuring the Password Policy Using the Console," on page 254
- "Configuring the Password Policy Using the Command-Line," on page 255

After configuring your password policy, we recommend that you configure an account lockout policy. For more information about configuring an account lockout policy, refer to "Configuring the Account Lockout Policy," on page 260.

Configuring the Password Policy Using the Console

To set up or modify the password policy for your Directory Server:

1. On the Directory Server Console, select the Configuration tab and then the Data node.
2. Select the Passwords tab in the right pane.

   This tab contains the password policy for the Directory Server.
3. You can specify that users must change their password the first time they log on by selecting the "User must change password after reset" checkbox.

   If you select this checkbox, only the Directory Manager is authorized to reset the users's password (using the field described in step 9). A regular administrative user cannot force the user to update their password.

4. To specify that users can change their own passwords, select the “User may change password” checkbox.
5. You can specify that users cannot change their password for a specific time by entering the number of days in the “Allow changes in X day(s)” text box.
6. To configure the server to maintain a history list of passwords used by each user, select the “Keep password history” checkbox. Specify the number of passwords you want the server to keep for each user in the “Remember X passwords” text box.
7. If you do not want user passwords to expire, select the “Password never expires” radio button.
8. If you want users to have 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.
9. If you have turned selected the “Password expire after X days” radio button, you need to specify how long before the password expires to send a warning to the user.In the “Send Warning X Days Before Password Expires” text enter the number of days before password expiration to send a warning.
10. If you want the server to check the syntax of a user password to make sure it meets the minimum requirements set by the password policy, select the “Check Password Syntax” checkbox. Then, specify the minimum acceptable password length in the “Password Minimum Length” text box.
11. Specify what encryption method you want the server to use when storing passwords from the “Password Encryption” pull-down menu.

    For detailed information about the encryption methods, refer to the `passwordStorageScheme` attribute in Table 7-1 on page 256.

    The “Password Encryption” menu might contain other encryption methods, as the directory dynamically creates the menu depending upon the existing encryption methods it finds in your directory.
12. When you have finished making changes to the password policy, click Save.

Configuring the Password Policy Using the Command-Line

This section describes the attributes you set to create a password policy for your server. Use ldapmodify to change these attributes in the `cn=config` entry.

The following table describes the attributes you can use to configure your password policy:

Table 7-1 Password Policy Attributes

Attribute Name	Definition
`passwordMustChange`	When `on`, this attribute requires users to change their passwords when they first login to the directory or after the password is reset by the Directory Manager. When `on`, the user is required to change their password even if user-defined passwords are disabled. If you choose to set this attribute to off, passwords assigned by the Directory Manager should not follow any obvious convention and should be difficult to discover. This attribute is `off` by default.
`passwordChange`	When `on`, this attribute indicates that users may change their own password. Choosing for users to set their own passwords runs the risk of users choosing passwords that are easy to remember. However, setting good passwords for the user requires a significant administrative effort. In addition, providing passwords to users that are not meaningful to them runs the risk that users will write the password down somewhere that can be discovered. This attribute is `on` by default.
`passwordExp`	When `on`, this attribute indicates that the user's password will expire after an interval given by the `passwordMaxAge` attribute. Making passwords expire helps protect your directory data because the longer a password is in use, the more likely it is to be discovered. This attribute is `off` by default.
`passwordMaxAge`	This attribute indicates the number of seconds after which user passwords expire. To use this attribute, you must enable password expiration using the `passwordExp` attribute. A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to 8640000 seconds (100days).

Table 7-1 Password Policy Attributes

Attribute Name	Definition
`passwordWarning`	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 iPlanet Directory Express and the Directory Server Gateway provide this functionality. By default, the directory sends the warning 86400 seconds (1day) before the password is about to expire. However, a password never expires until the warning message has been set. Therefore, if users don't bind to the Directory Server for longer than the passwordMaxAge, they will still get the warning message in time to change their password.
`passwordCheckSyntax`	When on, this attribute indicates that the password syntax will be checked by the server before the password is saved. Password syntax checking ensures that the password string meets or exceeds the minimum password length requirements and that the string does not contain any "trivial" words. A trivial word is any value stored in the `uid`, `cn`, `sn`, `givenName`, `ou`, or `mail` attributes of the user's entry. This attribute is `off` by default.
`passwordMinLength`	This attribute specifies the minimum number of characters that must be used in passwords. Shorter passwords are easier to crack. You can require passwords that are 2 to 512 characters long. Generally, a length of 6 to 8 characters is long enough to be difficult to crack but short enough for users to remember without writing it down. This attribute is set to 6 by default.

Table 7-1 Password Policy Attributes

Attribute Name	Definition
`passwordMinAge`	This attribute indicates the number of seconds that must pass before a user can change their password. Use this attribute in conjunction with the `passwordInHistory` attribute to discourage users from reusing old passwords. For example, setting the minimum password age to 2 days prevents users from repeatedly changing their passwords during a single session to cycle through the password history and reuse an old password once it has been removed from the history list. You can specify from 0 to 2147472000 seconds (24,855 days). A value of zero indicates that the user can change the password immediately. The default value of this attribute is `0`.
`passwordHistory`	This attribute indicates whether the directory stores a password history. When set to on, the directory stores the number of passwords you specify in the `passwordInHistory` attribute in a history. If a user attempts to reuse one of the password, the password will be rejected. When you set this attribute to `off`, any passwords stored in the history remain there. When you set this attribute back to `on`, users will not be able to reuse the passwords recorded in the history before you disabled the attribute. This attribute is `off` by default, meaning users can reuse old passwords.
`passwordInHistory`	This attribute indicates the number of passwords the directory stores in the history. You can store from 2 to 24 passwords in the history. This feature is not enabled unless the `passwordHistory` attribute is set to `on`. This attribute is set to 6 by default.

Table 7-1 Password Policy Attributes

Attribute Name	Definition
`passwordStorageScheme`	This attribute specifies the type of encryption used to store Directory Server passwords. The following encryption types are supported by Directory Server: • `SSHA` (Salted Secure Hash Algorithm). This method is recommends as it is the most secure. This is the default method. • `SHA` (Secure Hash Algorithm). A one-way hash algorithm that is the default encryption schema in Directory Server 4.x. • `crypt.` The UNIX crypt algorithm, provided for compatibility with UNIX passwords. • `clear`. This encryption type indicates that the password will appear in plain text. Note that passwords stored using crypt, SHA, or SSHA formats cannot be used for secure login through SASL Digest MD5. If you want to provide your own customized storage scheme, consult iPlanet Professional Services.

Setting User Passwords

An entry can be used to bind to the directory only if it has a `userpassword` attribute and if it has not been inactivated. Because user passwords are stored in the directory, you can use whatever LDAP operation you normally use to update the directory to set or reset the user passwords.

For information on creating and modifying directory entries, see Chapter 2, "Creating Directory Entries." For information on inactivating user accounts, refer to"Inactivating Users and Roles," on page 263.

You can also use the Users and Groups area of the Administration Server or the Directory Server Gateway to set or reset user passwords. For information on how to use the Users and Groups area, see the online help that is available in the Administration Server. For information on how to use the Gateway to create or modify directory entries, see the online help that is available in the Gateway.

Configuring the Account Lockout Policy

The lockout policy works in conjunction with the password policy to provide further security. The account lockout feature protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. You can set up your password policy so that a specific user is locked out of the directory after a given number of failed attempts to bind.

Configuring the account lockout policy is described in the following sections:

- "Configuring the Account Lockout Policy Using the Console," on page 260
- "Configuring the Account Lockout Policy Using the Command Line," on page 260

Configuring the Account Lockout Policy Using the Console

To set up or modify the account lockout policy for your Directory Server:

1. On the Directory Server Console, select the Configuration tab and then the Data node.
2. Select the Account Lockout tab in the right pane.
3. To enable account lockout, select the "Accounts may be locked out" checkbox.
4. Enter the maximum number of allowed bind failures in the "Lockout account after X login failures" text box. The server locks out users who exceed the limit you specify here.
5. Enter the number of minutes you want the server to wait before resetting the bind failure counter to 0 in the "Reset failure counter after X minutes" text box.
6. Set the interval you want users to be locked out of the directory.

   Select the Lockout Forever radio button to lock users out until their passwords have been reset by the administrator.

   Set a specific lockout period by selecting the Lockout duration radio button and entering the time (in minutes) in the text box.
7. When you have finished making changes to the account lockout policy, click Save.

Configuring the Account Lockout Policy Using the Command Line

This section describes the attributes you set to create an account lockout policy to protect the passwords stored in your server. Use ldapmodify to change these attributes in the `cn=config` entry.

The following table describes the attributes you can use to configure your account lockout policy:

Table 7-2 Account Lockout Policy Attributes

Attribute Name	Definition
`passwordLockout`	This attribute indicates whether users are locked out of the directory after a given number of failed bind attempts. You set the number of failed bind attempts after which the user will be locked out using the `passwordMaxFailure` attribute. You can lock users out for a specific time or until an administrator resets the password. This attribute is set to `off` by default, meaning that users will not be locked out of the directory.
`passwordMaxFailure`	This attribute indicates the number of failed bind attempts after which a user will be locked out of the directory. This attribute takes affect only if the `passwordLockout` attribute is set to `on`. This attribute is set to 3 bind failures by default.
`passwordLockoutDuration`	This attribute indicates the time, in seconds, that users will be locked out of the directory. You can also specify that a user is lock out until their password is reset by an administrator using the `passwordUnlock` attribute. By default, the user is locked out for 3600 second.

Table 7-2 Account Lockout Policy Attributes

Attribute Name	**Definition**
`passwordResetFailureCount`	This attribute specifies the time in seconds after which the password failure counter will be reset. Each time an invalid password is sent from the user's account, the password failure counter is incremented. If the `passwordLockout` attribute is set to on, users will be locked out of the directory when the counter reaches the number of failures specified by the `passwordMaxFailure` attribute. The account is locked out for the interval specified in the `passwordLockoutDuration` attribute, after which time the failure counter is reset to zero (0). Because the counter's purpose is to gauge when a hacker is trying to gain access to the system, the counter must continue for a period long enough to detect a hacker. However, if the counter was to increment indefinitely over days and weeks, valid users might be locked out inadvertently. The reset password failure count attribute is set 600 seconds by default.

Managing the Password Policy in a Replicated Environment

Password and account lockout policies are enforced in a replicated environment as follows:

- Password policies are enforced on the data master.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy information in your directory is replicated. The replicated attributes are:

- `passwordMinAge` and `passwordMaxAge`
- `passwordExp`
- `passwordWarning`

However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications. Account lockout counters and tiers are not replicated either.

When configuration 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 rebind, they may find that the bind fails until the replica registers the changes.
- You want the same bind behavior to occur on all servers, including masters and replicas. Make sure to create the same password policy configuration information on each server.
- Account lockout counters many not work as expected in a multi-mastered environment.
- Entries that are created for replication (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the `passwordExpirationTime` attribute to the entry and give it a value of `20380119031407Z` (the top of the valid range).

Inactivating Users and Roles

You can temporarily inactivate a single user account or a set of accounts. Once inactivated, a user cannot bind to the directory. The authentication operation will fail.

Users and roles are inactivated using the operational attribute `nsAccountLock`. When an entry contains the `nsAccountLock` attribute with a value of `true`, the server rejects the bind.

You use the same procedures for inactivating users and roles. However, when you inactivate a role, you are inactivating the members of the role and not the role entry itself. For more information about roles in general and how roles interact with access control in particular, refer to Chapter 5, “Advanced Entry Management.”

The rest of this section describes the following procedures:

- “Inactivating User and Roles Using the Console,” on page 264
- “Inactivating User and Roles Using the Command Line,” on page 264
- “Activating User and Roles Using the Console,” on page 265

- "Activating User and Roles Using the Command Line," on page 266

CAUTION You cannot inactivate the root entry (the entry corresponding to the root or sub suffix) on a database.

For more information on creating the entry for a root or sub suffix, refer to Chapter 2, "Creating Directory Entries" for more information. For more information on creating root and sub suffixes, refer to Chapter 3, "Configuring Directory Databases".

Inactivating User and Roles Using the Console

The following procedure describes inactivating a user or a role using the console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane and double-click the user or role you want to inactivate.

   The Edit Entry dialog box appears.

   You can also select Inactivate from the Object menu as a short cut.
3. Click Account in the left pane. The right pane states that the role or user is inactivated. Click Activate to activate the user or role.
4. Click OK to close the dialog box and save your changes.

   Once inactivated, you can view the state of the object by selecting Inactivation State from the View menu. The icon of the object then appears in the right pane of the console with a red slash through it.

Inactivating User and Roles Using the Command Line

To inactivate a user account, use the `ns-inactivate.pl` script. The following example describes using the `ns-inactivate.pl` script to inactivate Joe Frasier's user account:

```
ns-inactivate.pl -D "Directory Manager" -w secretpwd -p 389 -h
siroe.com -I "uid=jfrasier,ou=people,dc=siroe,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
-D	The DN of the directory administrator.
-w	The password of the directory administrator.
`-p`	Port used by the server.
-h	Name of the server on which the directory resides
-I	DN of the user account or role you want to inactivate.

For more information about running the `ns-inactivate.pl` script, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Activating User and Roles Using the Console

The following procedure describes activating a user or a role using the console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane and double-click the user or role you want to activate.

   The Edit Entry dialog box appears.

   You can also select Activate from the Object menu as a short cut.
3. Click Account in the left pane. The right pane states that the role or user is activated. Click Activate to activate the user or role.
4. If the user or role is a member of another inactivated role, the console displays an option for viewing the inactivated roles. Click Show Inactivated Roles to view the list of roles to which the user or role belongs.
5. Click OK when you are finished.

   Once reactivated, you can view the state of the object by selecting Inactivation State from the View menu. The icon of the role or user in the right pane of the console appears as normal. The red slash through the icon indicating it was inactive disappears.

Activating User and Roles Using the Command Line

To activate a user account, use the `ns-activate.pl` script. The following example describes using the `ns-activate.pl` script to activate Joe Frasier's user account:

```
ns-activate.pl -D "Directory Manager" -w secretpwd -p 389 -h
siroe.com -I "uid=jfrasier,ou=people,dc=siroe,dc=com"
```

The following table describes the `ns-inactivate.pl` options used in the example:

Option Name	Description
`-D`	The DN of the directory administrator.
`-w`	The password of the directory administrator.
`-p`	Port used by the server.
`-h`	Name of the server on which the directory resides
`-I`	DN of the user account or role you want to activate.

For more information about running the `ns-activate.pl` script, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Setting Resource Limits Based on the Bind DN

You can control server limits for search operations using special operational attribute values on the client application binding to the directory. You can set the following search operation limits:

Look through limit. Specifies how many entries can be examined for a search operation.

Size limit. Specifies the maximum number of entries the server returns to a client application in response to a search operation.

Time limit. Specifies the maximum time the server spends processing a search operation.

Idle timeout. Specifies the time a connection to the server can be idle before the connection is dropped.

NOTE The Directory Manager receives unlimited resources by default.

The resource limits you set for the client application takes precedence over the default resource limits you set for in the global server configuration.

This section gives procedures for the following:

- "Setting Resource Limits Using the Console," on page 267
- "Setting Resource Limits Using the Command Line," on page 267

Setting Resource Limits Using the Console

The following procedure describes setting resource limits for a user or a role using the console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left navigation pane and double-click the user or role for which you want to set resource limits.

   The Edit Entry dialog box appears.
3. Click Account in the left pane. The right pane contains the four limits you can set in the Resource Limits section.

   Entering a value of -1 indicates no limit.
4. Click OK when you are finished.

Setting Resource Limits Using the Command Line

The following operational attributes can be set for each entry using the command-line. Use `ldapmodify` to add the following attributes to the entry:

Attribute	Description
`nsLookThroughLimit`	Specifies how many entries examined for a search operation. Specified as a number of entries. Giving this attribute a value of -1 indicates that there is no limit.

Attribute	Description
`nsSizeLimit`	Specifies the maximum number of entries the server returns to a client application in response to a search operation. Giving this attribute a value of -1 indicates that there is no limit.
`nsTimeLimit`	Specifies the maximum time the server spends processing a search operation. Giving this attribute a value of `-1` indicates that there is no time limit.
`nsIdleTimeout`	Specifies the time a connection to the server can be idle, before the connection is dropped. The value is given in seconds. Giving this attribute a value of -1 indicate that there is no limit.

For example, you might set the size limit for an entry by performing an `ldapmodify` as follows:

```
ldapmodify -h myserver -p 389 -D “cn=directory manager” -w secretpwd
dn: uid=bjensen,ou=people,dc=siroe,dc=com
changetype: modify
add:nsSizeLimit
nsSizeLimit: 500
```

The `ldapmodify` statement adds the nsSizeLimit attribute to Babs Jensen’s entry and gives it a search return size limit of 500 entries.

Chapter 8

Managing Replication

Replication is an important mechanism for extending your directory service beyond a single server configuration. This chapter describes the tasks to be performed on the supplier servers and the consumer servers to set up single master replication, multi-master replication, and cascading replication. This chapter includes the following topics:

- Replication Overview
- Replication Scenarios
- Configuring Single-Master Replication
- Configuring Multiple-Master Replication
- Configuring Cascading Replication
- Configuration Tips
- Detailed Procedures
- Removing the Change Log
- Initializing Consumers
- Forcing Replication Updates
- Replication over SSL
- Replication with Earlier Releases
- Using the Retro Change Log Plug-In
- Monitoring Replication Status
- Solving Common Replication Conflicts

For conceptual information on how you can use replication in your directory deployment, see the *iPlanet Directory Server Deployment Guide*.

Replication Overview

Replication is the mechanism by which directory data is automatically copied from one Directory Server to another. Using replication, you can copy entire directory trees or subtrees between servers. Updates of any kind—entry additions, modifications, or even deletions—are automatically mirrored to other Directory Servers using replication.

Read-Write Replica/Read-Only Replica

A database that participates in replication is defined as a *replica*. There are two kinds of replicas: read-write or read-only. A read-write replica contains master copies of directory information and can be updated. A read-only replica refer all update operations to read-write replicas. A server can hold any number of read-only or read-write replicas.

Supplier/Consumer

A server that holds a replica that is copied to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is copied from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica. There are exceptions to this statement:

- In the case of cascading replication, the hub supplier holds a read-only replica that it supplies to consumers. For more information, refer to "Cascading Replication," on page 276.
- In the case of multi-master replication, both masters are suppliers and consumers for the same read-write replica. For more information, refer to "Multi-Master Replication," on page 274.

In iPlanet Directory Server 5.0, replication is always initiated by the supplier server, never by the consumer. This operation is called supplier-initiated replication. It allows you to configure a supplier server to push data to one or more consumer servers.

Earlier versions of the iPlanet Directory Server allowed consumer-initiated replication where you could configure consumer servers to pull data from a supplier server.

Change Log

Every supplier server maintains a change log. A change log is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers, or on other masters in the case of multi-master replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the change log.

In iPlanet Directory Server 5.0, the format of the change log has changed. It is now intended only for internal use by the server. If you have applications that need to read the change log, you can use the Retro Change log plug-in for backward compatibility. For more information, refer to “Using the Retro Change Log Plug-In,” on page 308.

Unit of Replication

In iPlanet Directory Server 5.0, the smallest unit of replication is a database. This means that you can replicate an entire database, but not a subtree within a database. Therefore, when you create your directory tree, you must take your replication plans into consideration. For more information on how to set up your directory tree, refer to *iPlanet Directory Server Deployment Guide*.

The replication mechanism also requires that one database correspond to one suffix. This means that you cannot replicate a suffix (or namespace) that is distributed over two or more databases using custom distribution logic. For more information on this topic, refer to “Creating and Maintaining Databases,” on page 81.

Replication Identity

When replication occurs between two servers, the replication process uses a special entry, often referred to as *Replication Manager* to identify replication protocol exchanges. The Replication Manager entry, or any entry you create to fulfill that role, has the following characteristics:

- You must create the entry on every server that receives updates from another server (a hub supplier or a dedicated consumer).
- When you configure a replica that receives updates from another server, you must specify this entry as the one authorized to perform replication updates.

- On the supplier server, when you configure the replication agreement, you must specify the DN of this entry.
- This entry must not be part of the replicated database
- This entry bypasses all access control rules defined on the consumer server.

NOTE In the Directory Server Console, this entry is referred to as the *supplier bind DN*. This can be misleading because the entry does not exist on the supplier server but on the consumer server.

For more information on creating the Replication Manager entry, refer to "Creating the Bind DN for Suppliers," on page 291.

Replication Agreement

Directory Servers use replication agreements to define replication. A replication agreement describes replication between *one* supplier and *one* consumer only. The agreement is configured on the supplier server. It identifies:

- The database to replicate
- The consumer server to which the data is pushed
- The times during which replication can occur
- The DN and credentials that the supplier server must use to bind (called the Replication Manager, or *supplier bind DN*)
- How the connection is secured (SSL, client authentication)

Compatibility with Earlier Versions of Directory Server

The replication mechanism in iPlanet Directory Server 5.0 is different from the mechanism used in earlier versions of Directory Server. Compatibility is provided in the following manner:

- Through the Legacy Replication Plug-in
- Through the Retro Changelog Plug-in

The legacy replication plug-in makes Directory Server 5.0 behave as a 4.x directory server in a consumer role. For information on how to implement legacy replication using this plug-in, refer to “Replication with Earlier Releases,” on page 306.

The retro change log plug-in can be used when you want a Directory Server 5.0 supplier to maintain a 4.x style change log. This is sometimes necessary for applications that have a dependency on the Directory Server 4.x change log format, because they read information from the change log. For more information on the retro change log plug-in, refer to “Using the Retro Change Log Plug-In,” on page 308.

Replication Scenarios

This section describes the most commonly used replication scenarios:

- Single master replication
- Multi-master replication
- Cascading replication

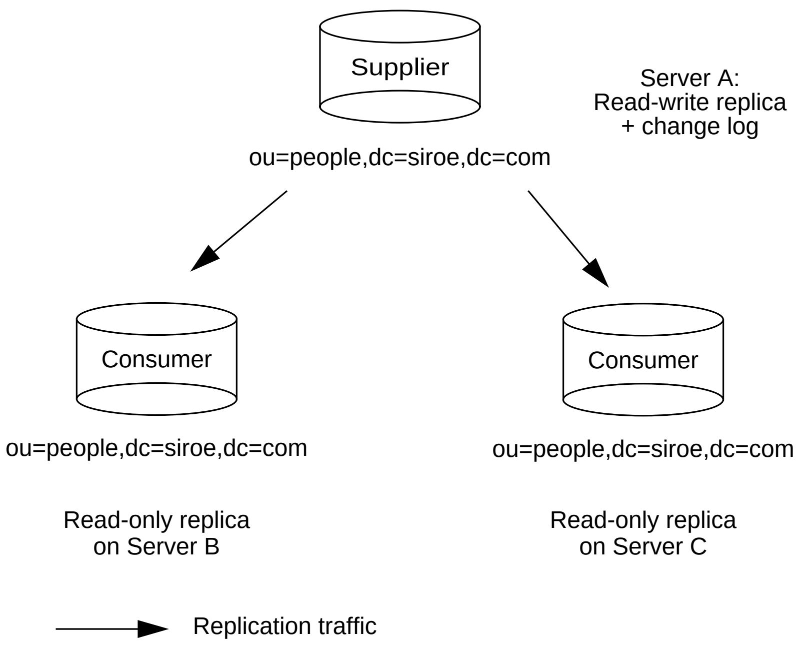
You can combine these basic scenarios to build the replication environment that best suits your needs.

NOTE Whatever replication scenario you choose to implement, remember to consider schema replication. For details, refer to *iPlanet Directory Server Deployment Guide*.

Single-Master Replication

In the most simple replication scenarios, the master copy of directory data is held in a single read-write replica on one server. The server also maintains a change log for this replica. You can have as many read-only replicas as you like. Such scenarios are called single-master configurations. Figure 8-1 shows an example of single-master replication.

Figure 8-1 Single-Master Replication



The `ou=people,dc=siroe,dc=com` suffix receives a large number of search requests. Therefore, to distribute the load, this tree which is mastered on Server A, is replicated to two read-only replicas located on Server B and Server C.

For information on setting up a single-master replication environment, refer to "Configuring Single-Master Replication," on page 277.

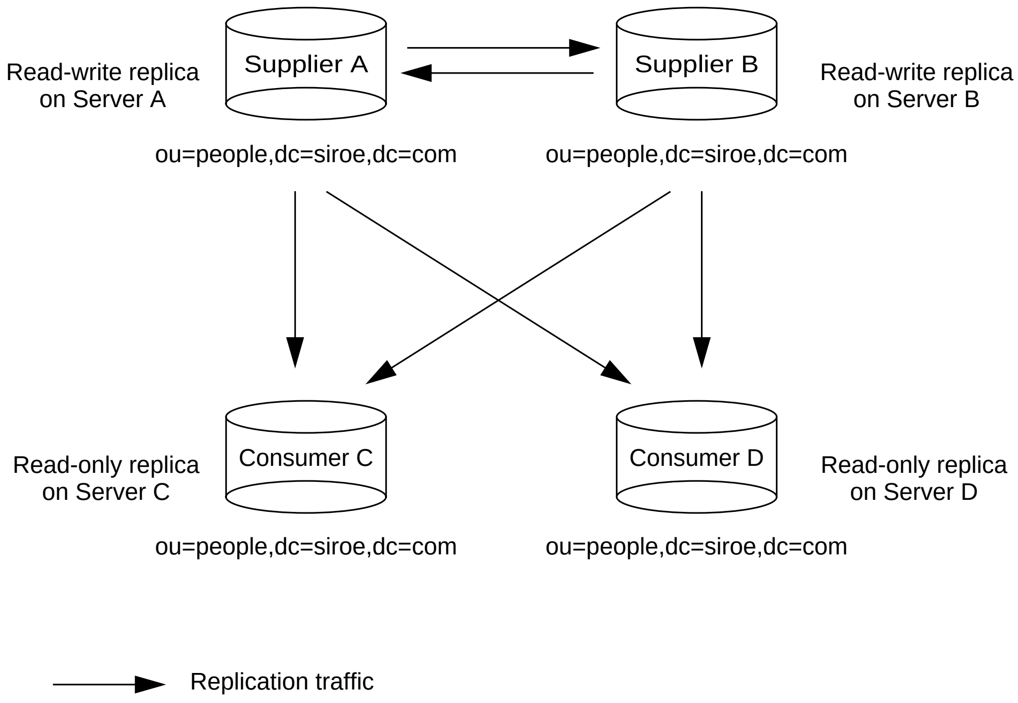
Multi-Master Replication

iPlanet Directory Server 5.0 also supports complex replication scenarios in which the same subtree can be mastered on two servers. This subtree is held in a read-write replica on each server. This means that each server maintains a change log for the 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 both suppliers. The consumers also have referrals defined for both suppliers which are used to forward any update requests that they receive. Such scenarios are called multi-master configurations.

Figure 8-2 shows an example of multi-master replication scenario.

Figure 8-2 Multi-Master Replication


Read-write replica on Server A
Supplier A
Supplier B
Read-write replica on Server B
ou=people,dc=siroe,dc=com
ou=people,dc=siroe,dc=com
Read-only replica on Server C
Consumer C
Consumer D
Read-only replica on Server D
ou=people,dc=siroe,dc=com
ou=people,dc=siroe,dc=com
Replication traffic

Multi-master configurations have the following advantages:

- Automatic write failover when one master/supplier is inaccessible
- Updates are made on the local supplier in a geographically distributed environment

NOTE Replication, especially multi-master replication, works better over high speed links than over slow links such as a WAN used in geographically distributed environments.

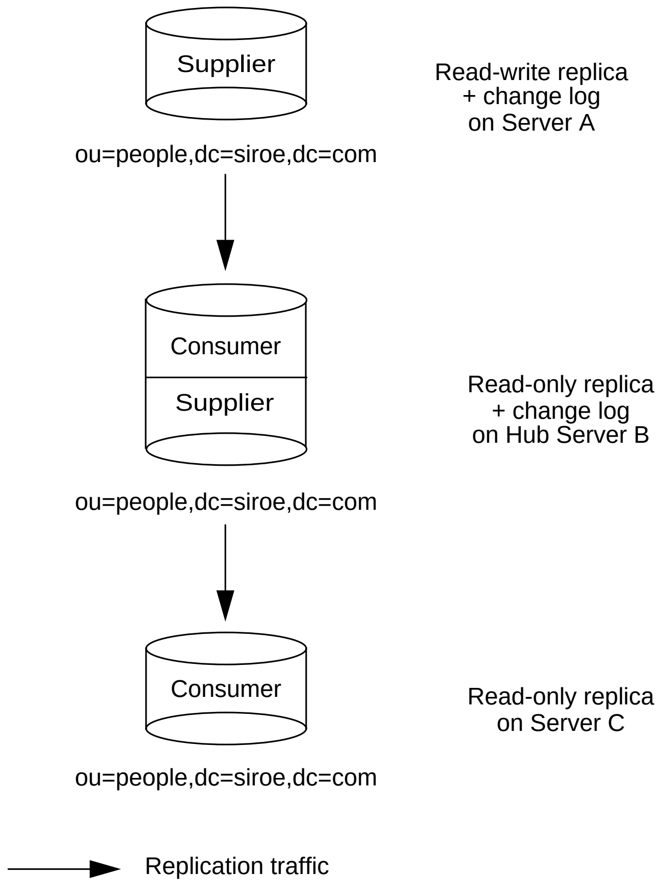
For information on setting up multi-master replication with two supplier servers and two consumer servers, refer to “Configuring Multiple-Master Replication,” on page 280.

Cascading Replication

In a cascading replication scenario, one server, often called a *hub supplier*, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a change log. It receives updates from the supplier server that holds the master copy of the data, and in turn supplies those updates to the consumer.

Figure 8-3 shows an example of cascading replication. This example shows a simple cascading replication scenario. You can create more complex scenarios with several hub suppliers.

Figure 8-3 Cascading Replication


Supplier
Read-write replica + change log on Server A
ou=people,dc=siroe,dc=com
Consumer
Supplier
Read-only replica + change log on Hub Server B
ou=people,dc=siroe,dc=com
Consumer
Read-only replica on Server C
ou=people,dc=siroe,dc=com
Replication traffic

For information on setting up cascading replication, refer to "Configuring Cascading Replication," on page 285.

NOTE You can combine multi-master and cascading replication. For example, in the mult-master scenario illustrated in Figure 8-2 on page 275, Server C and Server D could be hub suppliers that would replicated to any number of consumer servers.

Configuring Single-Master Replication

This section provides information on configuring single-master replication. The steps described in this chapter provide a high level outline of the procedure you need to follow. Cross-references to the detailed task description are provided at each step.

Configuring the Read-Only Replica on the Consumer Server

1. Create the database for the read-only replica, if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 72.

2. Create the entry corresponding to the supplier bind DN, if it does not exist. This is the special entry that the supplier will use to bind.

   a. In the Directory Server Console, click the Directory tab, and create an entry. For example you could use `cn=Replication Manager,cn=config`.

   b. Specify a userPassword attribute-value pair.

   c. If you have enabled password expiration, or intend to do so in future, disable the password expiration policy on this attribute, by adding the passwordExpirationTime attribute with a value of 20380119031407Z.

NOTE This entry must not be part of the replicated database.

3. Specify the replication parameters required for a read-only replica.

   a. In the Directory Server Console, click the Configuration tab.

b. In the navigation tree, expand the Replication folder, and highlight the replica database.

The Replica Settings tab is displayed in the right-hand side of the window.

c. Check the Enable Replica checkbox.

d. In the Replica Role section, select the Dedicated Consumer radio button.

e. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

f. Specify a purge delay.

This option indicates how often the state information stored in the replicated entries is purged.

g. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

h. Specify any supplier servers to which you want to refer updates.

By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

Automatic referrals assume that clients will bind over a regular connection, and therefore, are of the form `ldap://`*servername*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*servername*`:`*port*.

4. Click Save to save the replication parameters for the replica.

Repeat these steps for every read-only replica in your replication configuration.

Configuring the Read-Write Replica on the Supplier Server

1. Specify the supplier settings for the server.

   a. In the Directory Server Console, click the Configuration tab.

   b. In the navigation tree, highlight the Replication node.

   c. In the right-hand side of the window, click the Supplier Settings tab.

   d. Check the Enable Change Log checkbox.

      This activates all of the fields in the pane below that were previously greyed out.

   e. Specify a change log by clicking the Use Default button, or click the Browse button to display a file selector.

   f. Set the change log parameters (number and age).

      You must clear the unlimited checkboxes if you want to specify different values.

   g. Click Save to save the supplier settings.

2. Specify the replication parameters required for a read-write replica.

   a. In the navigation tree on the Configuration tab, expand the Replication node and highlight the database to replicate.

      The Replica Settings tab is displayed in the right-hand side of the window.

   b. Check the Enable Replica checkbox.

   c. In the Replica Role section, select the Single Master radio button.

   d. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

      The replica ID must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

   e. Specify a purge delay.

      This option indicates how often the state information stored in the replicated entries is purged.

   f. Click Save to save the replication parameters for the database.

3. Create a replication agreement.

   You must create one replication agreement for each read-only replica. For example, in the case illustrated in Figure 8-1 on page 274, Server A holds two replication agreements, one for Server B, and one for Server C.

   a. In the navigation tree on 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. This will start the Replication Agreement Wizard.

   b. Go through the steps in the replication wizard by clicking Next to move to the following step.

      You can initialize the read-only replicas from the Replication Agreement Wizard, or at anytime afterwards. For information on initializing read-only replicas, refer to "Initializing Consumers," on page 297.

   When you have finished, the replication agreement is set up.

Configuring Multiple-Master Replication

In a multi-master replication configuration, two suppliers can accept updates, synchronize with each other, and update all consumers. The consumers can send referrals for updates to both masters.

To set up multi-master replication such as the configuration shown in Figure 8-2 on page 275, between two suppliers Server A and Server B that each hold a read-write replica, and two consumers Server C and Server D that each hold a read-only replica, you must perform the following tasks:

Configuring the Read-Only Replicas on the Consumer Servers

Perform these steps on each consumer server:

1. Create the database for the read-only replica, if it does not exist.

   For instructions, refer to "Creating Suffixes," on page 72.

2. Create the entry corresponding to the supplier bind DN, if it does not exist. This is the special entry that the supplier will use to bind.

a. In the Directory Server Console, click the Directory tab, and create an entry. For example you could use `cn=Replication Manager,cn=config`.

b. Specify a userPassword attribute-value pair.

c. If you have enabled password expiration, or intend to do so in future, disable the password expiration policy on this attribute, by adding the passwordExpirationTime attribute with a value of 20380119031407Z.

NOTE This entry must not be part of the replicated database.

---

3. Specify the replication parameters required for a read-only replica.

a. In the Directory Server Console, click the Configuration tab.

b. In the navigation tree, expand the Replication folder, and highlight the replica database.

The Replica Settings tab is displayed in the right-hand side of the window.

c. Check the Enable Replica checkbox.

d. In the Replica Role section, select the Dedicated Consumer radio button.

e. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

f. Specify a purge delay.

This option indicates how often the state information stored in the replicated entries is purged.

g. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

   h. Specify any supplier servers to which you want to refer updates.

      By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

      Automatic referrals assume that clients will perform a simple bind, and therefore, are of the form `ldap://`*servername*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*servername*`:`*port*.

4. Click Save to save the replication parameters for the replica.

Repeat these steps for every read-only replica in your replication configuration.

Configuring the Read-Write Replicas on the Supplier Servers

Perform these steps on each supplier server:

1. On Server A and Server B, specify the supplier settings for each server.

   a. In the Directory Server Console, click the Configuration tab.

   b. In the navigation tree, highlight the Replication node.

   c. In the right-hand side of the window, click the Supplier Settings tab.

   d. Check the Enable Change Log checkbox.

      This activates all of the fields in the pane below that were previously greyed out.

   e. Specify a change log by clicking the Use Default button, or click the Browse button to display a file selector.

   f. Set the change log parameters (number and age).

      You must clear the unlimited checkboxes if you want to specify different values.

   g. Click Save to save the supplier settings.

2. Create the entry corresponding to the supplier bind DN, if it does not exist. This entry is necessary so that the two masters can replicate changes to one another.

   a. In the Directory Server Console, click the Directory tab, and create an entry. For example you could use `cn=Replication Manager,cn=config`.

   b. Specify a userPassword attribute-value pair.

   c. If you have enabled password expiration, or intend to do so in future, disable the password expiration policy on this attribute, by adding the passwordExpirationTime attribute with a value of 20380119031407Z.

NOTE This entry must not be part of the replicated database.

3. On Server A and Server B, specify the replication parameters for the multi-mastered read-write replica.

   a. In the navigation tree on the Configuration tab, expand the Replication node and highlight the database to replicate.

   The Replica Settings tab is displayed in the right-hand side of the window.

   b. Check the Enable Replica checkbox.

   c. In the Replica Role section, select the Multiple Master radio button.

   d. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

   The replica ID must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

   e. Specify a purge delay.

   This option indicates how often the state information stored in the replicated entries is purged.

   f. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

g. Specify the supplier server to which you want to refer updates (the other master in the multi-master set).

   Only specify the URL for the supplier server if you want clients to bind using SSL. In such a case, you must specify a URL beginning with `ldaps://`.

h. Click Save to save the replication parameters for the database.

4. On Server A, set up the following replication agreements:

   ❍ One with supplier Server B, where B is configured as a *consumer* for the replica.

   ❍ One for each consumer, Server C and Server D.

   a. In the navigation tree on 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. This will start the Replication Agreement Wizard.

   b. Go through the steps in the replication wizard by clicking Next to move to the following step.

      You can initialize the read-only replicas and the read-write replica on Server B from the Replication Agreement Wizard, or at anytime afterwards. For information on the order and procedure for initializing read-only replicas, refer to"Initializing the Replicas," on page 285, and "Initializing Consumers," on page 297.

   When you have finished, the replication agreement is set up.

5. On Server B, set up the following replication agreements:

   ❍ One with supplier Server A, where A is declared as a *consumer* for the replica. During this operation, do not initialize Server A from Server B if you have already initialized Server B from Server A in Step 4.

   ❍ One for each consumer, Server C and Server D.

NOTE This means that Server A and Server B have mutual replication agreements so that they can accept updates from each other.

When you have configured the servers holding the read-write replicas, the necessary replication agreements, and the servers holding the read-only replicas, you are ready to initialize replication. You can perform this task when you create the replication agreements on the supplier servers, or at any time afterwards.

Initializing the Replicas

In the case of multi-master replication, you should initialize replicas in the following order:

1. Ensure one master has the complete set of data to replicate. Use this master to initialize the replica on the other master in the multi-master replication set.
2. Initialize the replicas on the consumer servers from any one of the two masters.

For information on initializing replicas, refer to "Initializing Consumers," on page 297.

Configuring Cascading Replication

This section provides information on setting up cascading replication.

Configuring the Read-Write Replica on the Supplier Server

Perform these steps on the supplier server that holds the original copy of the database:

1. Specify the supplier settings for the server.
   a. In the Directory Server Console, click the Configuration tab.
   b. In the navigation tree, highlight the Replication node.
   c. In the right-hand side of the window, click the Supplier Settings tab.
   d. Check the Enable Change Log checkbox.

      This activates all of the fields in the pane below that were previously greyed out.

   e. Specify a change log by clicking the Use Default button, or click the Browse button to display a file selector.
   f. Set the change log parameters (number and age).

      You must clear the unlimited checkboxes if you want to specify different values.
   g. Click Save to save the supplier settings.

2. Specify the required replication parameters.
   a. In the navigation tree on the Configuration tab, expand the Replication node and highlight the database to replicate.

      The Replica Settings tab is displayed in the right-hand side of the window.
   b. Check the Enable Replica checkbox.
   c. In the Replica Role section, select the Single Master radio button.
   d. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

      The replica ID must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.
   e. Specify a purge delay.

      This option indicates how often the state information stored in the replicated entries is purged.
   f. Click Save to save the replication parameters for the database.

Configuring the Read-Only Replica on the Hub Supplier

Perform these steps on the hub supplier that receives replication updates from the master, and propagates them to consumers:

1. Create the database for the replica, if it does not exist.

   For instructions, refer to “Creating Suffixes,” on page 72.
2. Create the entry corresponding to the supplier bind DN, if it does not exist. This is the special entry that the supplier will use to bind.

   a. In the Directory Server Console, click the Directory tab, and create an entry. For example you could use `cn=Replication Manager,cn=config`.

   b. Specify a userPassword attribute-value pair.

   c. If you have enabled password expiration, or intend to do so in future, disable the password expiration policy on this attribute, by adding the passwordExpirationTime attribute with a value of 20380119031407Z.

NOTE This entry must not be part of the replicated database.

3. Specify the required replication parameters.

   a. In the navigation tree on the Configuration tab, expand the Replication node and highlight the database to replicate.

   The Replica Settings tab is displayed in the right-hand side of the window.

   b. Check the Enable Replica checkbox.

   c. In the Replica Role section, select the Hub radio button.

   d. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

   You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

   e. Specify a purge delay.

   This option indicates how often the state information stored in the replicated entries is purged.

   f. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

g. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will bind over a regular connection, and therefore, are of the form `ldap://`*servername*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*servername*`:`*port*.

h. Click Save to save the replication parameters for the database.

Configuring the Read-Only Replica on the Consumer Server

4. On the consumer server, create the database for the replica if it does not exist.

   For instructions, refer to “Creating Suffixes,” on page 72.

5. On the consumer server, create the entry corresponding to the supplier bind DN, if it does not exist.

6. On the consumer server, specify the replication parameters for the read-only replica.

   a. In the Directory Server Console, click the Configuration tab.

   b. In the navigation tree, expand the Replication folder, and highlight the replica database.

      The Replica Settings tab is displayed in the right-hand side of the window.

   c. Check the Enable Replica checkbox.

   d. In the Replica Role section, select the Dedicated Consumer radio button.

   e. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

      You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

   f. Specify a purge delay.

      This option indicates how often the state information stored in the replicated entries is purged.

g. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

h. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   In the case of cascading replication, referrals are automatically sent to the hub supplier, which in turn refers the request to the original master. Therefore, you should set a referral to the original master to replace the automatically generated referral.

7. On the supplier server, set up the replication agreement between this server and the hub supplier.

   a. In the navigation tree on 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. This will start the Replication Agreement Wizard.

   b. Go through the steps in the replication wizard by clicking Next to move to the following step.

      You can initialize the read-only replicas from the Replication Agreement Wizard, or at anytime afterwards. For information on initializing read-only replicas, refer to “Initializing Consumers,” on page 297.

8. On the hub supplier, set up the replication agreement between this server and the consumer.

When you have configured the replicas on each server, and the necessary replication agreements between servers, you can initialize the read-only replicas on the hub supplier, and on the consumer. You can perform this task from the replication agreement wizard while you are configuring the supplier server and the hub supplier server, or at any time afterwards.

Initializing the Replicas

In the case of cascading replication, you should initialize replicas in the following order:

1. Use the supplier server to initialize the replica on the hub supplier.
2. From the hub supplier, initialize the replica on the consumer.

For information on initializing replicas, refer to “Initializing Consumers,” on page 297.

Configuration Tips

If you are configuring replication for a large number of servers, and your configuration is relatively complex, for reasons of efficiency you should proceed in the following order:

1. On all consumer servers:
   - Create the replica databases
   - Create the Replication Manager entry
   - Specify the replica settings for a read-only replica
2. On all hub suppliers:
   - Create the replica databases
   - Create the Replication Manager entry
   - Specify the supplier settings for replication (includes change log configuration)
   - Specify the replica settings for a read-write replica
3. On all suppliers:
   - Create the replica databases
   - Specify the supplier settings for replication (includes change log configuration)
   - Specify the replica settings for a read-write replica
4. Configure replication agreements on all suppliers:
   - Between suppliers in a multi-master set

  - ❍ Between suppliers and dedicated consumers
  - ❍ Between suppliers and hub suppliers

  Optionally, you can initialize the replicas on the consumer servers at this stage.

5. Configure replication agreements on all hub suppliers, between the hub supplier and the dedicated consumers.

   Optionally, you can initialize the replicas on the consumer servers at this stage.

NOTE It is very important to create and configure all replicas before you attempt to create a replication agreement. This also means that when you create the replication agreement, you can choose to initialize consumers immediately.

Detailed Procedures

This section contains a description of the tasks you need to perform to configure replication.

Creating the Bind DN for Suppliers

A critical part of setting up replication is to create the entry that the suppliers will use to bind to the consumer servers to perform replication updates.

The supplier bind DN must meet the following criteria:

- It must be unique
- It must correspond to an actual entry on the consumer server
- It must not be part of the replicated database
- It must be defined in the replication agreement on the supplier server

For example, you could create an entry `cn=Replication Manager,cn=config` under the `cn=config` tree on the consumer server. This would be the supplier bind DN that all suppliers would use to bind to the consumer to perform replication operations.

To create the supplier bind entry:

1. On each server that acts as a consumer in replication agreements, create a special entry that the supplier will use to bind.

   This entry must not be part of the replicated database. For example, you could use `cn=Replication Manager,cn=config`. Make sure you create the entry with the attributes required by the authentication method you specify in the replication agreement.

2. Specify a `userPassword` attribute-value pair.
3. If you have enabled password expiration, or intend to do so in future, disable the password expiration policy on this attribute, by adding the passwordExpirationTime attribute with a value of 20380119031407Z.

When you configure a replica as a consumer, you must use the DN of this entry to define the supplier bind DN.

Configuring Supplier Settings

On any server that holds the master copy of a replica, you must specify supplier parameters.

To configure supplier settings:

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

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 26.

2. In the left navigation tree, highlight the Replication node.
3. In the right navigation window, click the Supplier Settings tab.
4. Check the Enable Change Log checkbox.

   This activates all of the fields in the pane below that were previously grayed out.

5. Specify a change log by clicking the Use Default button, or click Browse to display a file selector.
6. Set the change log number and age parameters.

   You must clear the unlimited checkboxes to specify different values.

7. Click Save to save the supplier settings.

Configuring a Read-Write Replica

For each read-write replica, you must specify the appropriate replication parameters.

To configure a read-write replica:

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

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 26.

2. In the left navigation tree, expand the Replication folder and highlight the database to replicate.

   The Replication tab is displayed in the right navigation window.

3. Check the Enable Replica checkbox.
4. In the Replica Role section, select the Single Master or Multi-Master radio button.
5. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

   The replica ID must be unique for a given suffix. Make sure you specify an ID that is different from the IDs used for read-write replicas on this server and on other servers.

6. Specify a purge delay.

   This option indicates how often the state information stored in the replicated entries is purged.

7. Click Save to save the replication parameters for the database.

Configuring a Read-Only Replica

For each read-only replica on the consumer server, you must specify the appropriate replication parameters.

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

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 26.

2. In the left navigation tree, expand the Replication folder, and highlight the replica database.

   The Replica Settings tab is displayed in the right navigation window.

3. Check the Enable Replica checkbox.

4. In the Replica Role section, select the Dedicated Consumer radio button.

5. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

   You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

6. Specify a purge delay.

   This option indicates how often the state information stored in the replicated entries is purged.

7. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

8. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients will bind over a regular connection, and therefore, are of the form `ldap://`*servername*`:`*port*. If you want clients to bind to the supplier using SSL, you can use this field to specify a referral of the form `ldaps://`*servername*`:`*port*.

   In the case of cascading replication, referrals are automatically sent to the hub supplier, which in turn refers the request to the original master. Therefore, you should set a referral to the original master to replace the automatically generated referral.

9. Click Save to save the replication parameters for the replica.

Configuring a Hub Supplier

In a cascading replication environment, configure the hub supplier as follows:

1. On the Directory Server Console, click the Configuration tab.

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 26.

2. In the left navigation tree, expand the Replication folder and highlight the database to replicate.

   The Replica Settings tab is displayed in the right navigation window.

3. Check the Enable Replica checkbox.

4. In the Replica Role section, select the Hub radio button.

5. In the Common Settings section, specify a Replica ID (an integer between 1 and 254 inclusive).

   You must specify the same replica ID as for the read-write replica that supplies updates to this replica. The replica ID must be unique for a given suffix.

6. Specify a purge delay.

   This option indicates how often the state information stored in the replicated entries is purged.

7. In the Replica Update Settings section, specify the bind DN that the supplier will use to bind to the replica.

   This bind DN should correspond to the entry created in Step 2. Note that the bind DN corresponds to a privileged user, because it is not subject to access control.

8. Specify any supplier servers to which you want to refer updates.

   By default, all updates are first referred to the supplier servers that you specify here. If you specify none, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   You can choose to either add the supplier servers that you specify to the automatically generated list, or to use the supplier servers that you specify to replace the automatically generated list of servers.

9. Click Save to save the replication parameters for the database.

Creating a Replication Agreement

This section explains how to create a replication agreement. You must create a replication agreement on the supplier server for each read-write replica that is supplied to a consumer server or a hub supplier.

Before you can create a replication agreement, you must:

- Configure supplier settings on the server, as described in “Configuring Supplier Settings,” on page 292.
- Configure replication parameters for suppliers, as described in “Configuring a Read-Write Replica,” on page 293.
- Configure replication parameters for hub suppliers (if any), and consumers, as described in “Configuring a Read-Only Replica,” on page 293.

To create a replication agreement:

1. On the Directory Server Console, click the Configuration tab.

   For information on starting the Directory Server Console, “Using the Directory Server Console,” on page 26.

2. In the navigation tree, expand the Replication folder, right-click the database to replicate, and select New Replication Agreement.

   Alternatively, highlight the database and select New Replication Agreement from the Object menu. This will start the Replication Agreement Wizard.

3. Go through the steps in the Replication Wizard by clicking Next to move to the following step.

   When you have finished, the replication agreement is set up. An icon representing the replication agreement is displayed under the database icon.

Removing the Change Log

NOTE If you remove the change log, you will need to reinitialize your consumer servers.

You can remove the change log using the Directory Server Console. On the supplier server:

1. On 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 Change Log checkbox.

   This deletes the change log.
4. Click Save.
5. Restart the Directory Server.
6. Reinitialize your consumers.

   See "Initializing Consumers," on page 297 for information.

You can actually remove an old change log without stopping to log changes by modifying the change log location. When you do this, a new change log is created in the directory you specify, and the old change log is deleted.

Initializing Consumers

There are two ways to initialize a consumer:

- Online Consumer Creation

  This method is the easiest to perform but is not the best option if you are replicating data over a slow connection.
- Manual Consumer Creation

  This is the more difficult but most efficient method.

This section first describes consumer initialization in detail and then provides instructions on both consumer creation methods.

When to Initialize a Consumer

After you have created a replication agreement, you must initialize the consumer. That is, you must physically copy directory data from the supplier server to the consumer server so that future changes can be replayed to consumer servers.

Consumer initialization involves copying a directory subtree from the supplier server to the consumer server. Once the subtree has been physically placed on the consumer, the supplier server can begin replaying update operations to the consumer server.

Under normal operations, the consumer should not ever have to be initialized again. However, if the data on the supplier server is restored from backup for any reason, then you should reinitialize all of the consumers supplied by it.

You can use one of the following methods to initialize a consumer:

- Online consumer creation
- Manual consumer creation

Online consumer creation is an effective method of initializing a small number of consumers. Each replica is initialized in sequence; therefore, this method is not adapted for initializing a large number of replicas.

Online consumer creation is the method used when you initialize the consumer while configuring the replication agreement on the supplier server.

Manual consumer creation is an effective method of initializing a large number of consumers from a single LDIF file.

The process that you use to initialize or reinitialize a consumer differs depending on the type of consumer creation you use. See "Online Consumer Creation," on page 298 or "Manual Consumer Creation," on page 299 for more information.

NOTE When a consumer server is being initialized via online consumer initialization, all operations (including searches) on the replica are referred to the supplier server until the initialization process is completed.

Online Consumer Creation

Online consumer creation is the easiest way to initialize or reinitialize a consumer. However, if you are replicating across a slow link, this process can be very time consuming, and you may find that manual consumer creation is a more appropriate approach (refer to "Manual Consumer Creation," on page 299 for more information).

How to Use Online Consumer Creation

This method assumes that you have already created a replication agreement, but that you have not yet initialized the consumer identified in the agreement. For details on creating replication agreements, see “Creating a Replication Agreement,” on page 296.

1. On the supplier server, on the Directory Server Console, select the Configuration tab.
2. Expand the Replication folder, then expand the replicated database. Right-click the replication agreement, and choose Initialize Consumer from the pop-up menu.

   A message is displayed to warn you that any information already stored in the replica on the consumer will be removed.
3. Click Yes in the confirmation box.

Online consumer creation begins immediately. You can check the status of the online consumer creation on the Summary tab in the Status box. If online consumer creation 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 creation finishes, the status changes to reflect this.

For more information about monitoring replication and initialization status, see “Monitoring Replication Status,” on page 311.

Manual Consumer Creation

Manual consumer creation is the fastest method of consumer initialization for sites that are replicating very large numbers of entries. However, the manual process is more complicated than the online creation process.

You should use the manual process whenever you find that the online process is inappropriate due to performance concerns.

To manually initialize or reinitialize a server:

1. Create a replication agreement.

   See “Creating a Replication Agreement,” on page 296.
2. Export the replica to an LDIF file.

   See “Exporting a Replica to LDIF,” on page 300.

3. Import the LDIF file to the consumer server.

See “Importing the LDIF File to the Consumer Server,” on page 300 for instructions.

Exporting a Replica to LDIF

You can convert the replica to LDIF:

- When you create a replication agreement by selecting “Create consumer initialization file” in the Initialize Consumer dialog box of the Replication Wizard.
- From the Directory Server Console, at any time, by right-clicking the replication agreement under the Replication folder, and choosing “Create LDIF File” from the pop-up menu.
- From the command line, by using the export command as described in “Exporting to LDIF From the Command Line,” on page 143.

Importing the LDIF File to the Consumer Server

Create the read-only replica on the consumer server from the LDIF file by using either the import features in the Directory Server Console, or the `ldif2db` scripts. These available methods are described in “Importing From the Command Line,” on page 137.

If you use these scripts, remember to bind using the supplier bind DN configured on the consumer server.

Forcing Replication Updates

When you stop a directory server involved in replication for regular maintenance, when it comes back online, you need to ensure that it gets updated by through replication immediately. In the case of a master in a multi-master environment, the directory information needs to be updated by the other master in the multi-master set. In other cases, when a hub supplier or a dedicated consumer is taken offline for maintenance, when they come back online, they need to be updated by the supplier server.

Note that if you have configured replication agreements to always keep the supplier server and the consumer server in sync, this is not sufficient to bring back up-to-date a server that has been offline for over five minutes. The reason is that with the "Always Keep in Sync" option, the server generates a replication operation for every update operation it processes. However, if this replication operation cannot be performed because the consumer is offline, the operation times out after 10 minutes.

NOTE The procedures described in this section can only be used when replication is already set up, *and* consumers have been initialized.

Several methods are available for ensuring that directory information will be synchronized immediately when a server comes back online:

- From the Directory Server Console on the supplier server that holds the reference copy of the directory information.
- By using a customizable script on the server that needs to be updated.

Forcing Replication Updates from the Console

To ensure that replication updates are sent immediately when a consumer or a master in a multi-master set comes back online after a period of time, you can perform these steps on the supplier server that holds the current version of the directory information:

1. On the Directory Server Console, click the Configuration tab, expand the Replication folder and the database nodes until you select the replication agreement corresponding to the replica that you must update.
2. Right click the replication agreement, and choose Send Updates Now from the drop-down list.

   This initiates replication toward the server that holds the information that needs to be updated.

Forcing Replication Updates from the Command Line

From the consumer that requires updating, you can run a script that prompts the supplier to send replication updates immediately. This script is shown in Code Example 8-1 on page 303.

You can copy this example and give it a meaningful name, for example, `replicate_now.sh`. You must provide actual values for the variables listed in Code Example 8-1.

Code Example 8-1 Replicate_Now Script

```
#!/bin/sh
SUP_HOST=supplier_hostname
SUP_PORT=supplier_portnumber
SUP_MGRDN=supplier_directoryManager
SUP_MGRPW=supplier_directoryManager_passwd
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.$$
```

If you intend to use this script, you must replace the variable parameters with actual values in your replication environment.

Table 8-1 Replicate_Now Variables

Variable	Definition
supplier_hostname	Hostname of the supplier to contact for information on replication agreements with the current consumer.
supplier_portnumber	LDAP port in use on the supplier.
supplier_directoryManager	DN of the privileged Directory Manager user on the supplier.
supplier_directoryManager_passwd	Password of the privileged Directory Manager user on the supplier.
consumer_hostname	Hostname of the current consumer.
consumer_portnumber	LDAP port in use on the consumer.

If you want the update operation to occur over an SSL connection, you must modify the `ldapmodify` command in the script with the appropriate parameters and values. For information on the `ldapmodify` command, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Replication over SSL

You can configure Directory Servers involved in replication so that all replication operations occur over an SSL connection.

To use replication over SSL, you must first do the following:

- Configure both your supplier and consumer servers to use SSL.
- Configure your consumer server to recognize your supplier server’s certificate as the supplier DN. You do this only if you want to use SSL client authentication rather than simple authentication.

These procedures are described in Chapter 11, “Managing SSL.”

NOTE Replication over SSL will fail in the following cases:

- If the supplier's certificate is a self-signed certificate
- If the supplier's certificate is an SSL server-only certificate, that is it can't act as a client during an SSL handshake.

When your servers are configured to use SSL, the following methods are available to ensure replication operations occur over SSL connections:

- Using the Replication Wizard, when you set up the replication agreement between two Directory Servers.
- From the Directory Server Console, at any time after the initial replication agreement was configured.

Configuring Replication Over SSL From the Replication Wizard

1. On the Directory Server Console of the supplier server, click the Configuration tab, expand the Replication folder and select the database that you want to replicate.
2. Right-click the database, and choose New Replication Agreement from the drop-down menu.

   The Replication Agreement Wizard is displayed.
3. Go through each step in the Replication Agreement Wizard until you reach the Source and Destination window.
4. In the Connection section, check "Using Encrypted SSL Connnection".
5. Select "SSL Client Authentication" or "Simple Authentication.

   If you select SSL Client Authentication, the supplier and consumer servers will use certificates to authenticate to each other.

   If you select Simple Authentication, the supplier and consumer servers will use a bind DN and password to authenticate to each other. You must specify this information in the text fields provided. When you specify this option, simple authentication takes place over a secure channel but without certificates.
6. Click Next, and proceed with the replication setup.

Configuring Replication Over SSL From the Console

1. On the Directory Server Console of the supplier server, click the Configuration tab, expand the Replication folder and select the replication agreement that you want to modify to enable replication over SSL.
2. Click the Connection tab in the right navigation window.

   This displays the replication connection parameters.
3. Select "SSL Client Authentication" or "Simple Authentication.

   If you select SSL Client Authentication, the supplier and consumer servers will use certificates to authenticate to each other.

   If you select Simple Authentication, the supplier and consumer servers will use a bind DN and password to authenticate to each other. You must specify this information in the text fields provided. When you specify this option, simple authentication takes place over a secure channel but without certificates.
4. Click Save.

Replication with Earlier Releases

This section provides information on how to optimize replication with earlier releases of iPlanet Directory Server. iPlanet Directory Server 5.0 can be involved in replication scenarios with earlier releases of Directory Server, providing the following conditions are met:

- Directory Server 5.0 is defined as a consumer in the replication agreement.
- The legacy suppliers can be Directory Server 4.0, 4.1, 4.11, and 4.12.

The following restrictions apply:

- A legacy Directory Server and a 5.0 Directory Server cannot update the same replica. However, a 5.0 Directory Server can have different replicas, where one is supplied by a legacy Directory Server, and the other is supplied by a 5.0 Directory Server.
- Directory Server 5.0 cannot be a supplier for other replicas.

The main advantage of being able to use a Directory Server 5.0 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 *iPlanet Directory Server Installation Guide*.

Configuring Directory Server 5.0 as a Legacy Consumer

If you intend to use your Directory Server 5.0 as a legacy consumer of an earlier release of Directory Server, you must configure it as follows:

1. On the Directory Server Console, click the Configuration tab.

   For information on starting the Directory Server Console, "Using the Directory Server Console," on page 26.

2. On the Configuration tab, select the Replication node and click the Legacy Consumer Settings tab in the right pane.

3. Check the Enable Legacy Consumer checkbox.

   This activates the fields in the Authentication box.

4. Specify the Supplier DN that the legacy supplier server will use to bind.

   Optionally, you can specify a password. The password must contain at least 8 characters.

5. Click Save.

   You must now configure legacy consumer settings for each replica that will receive updates from a legacy supplier.

6. In the navigation tree, expand the Replication node and select a replica that will receive updates from the legacy supplier.

7. On the Replication tab in the right pane, check the Enable Replication and the Enable Legacy Consumer checkboxes.

   These options are the only ones required for replication to work. You can optionally specify a Replica ID. It is not necessary to specify a Supplier DN, because the one you specified in Step 4 will be used.

8. Click Save.

Repeat Step 7 and Step 8 for each read-only replica that will receive updates from a legacy supplier.

To complete your legacy replication setup, you must now configure the legacy supplier to replicate to the 5.0 Directory Server. For instructions on configuring a replication agreement on a 4.x Directory Server, refer to the documentation for your legacy Directory Server.

NOTE Directory Server Console will not prevent you from configuring a database as a read-write replica and enabling legacy consumer settings. This makes migration easier because you can configure your 5.0 Directory Server as you want it to be after the migration, and activate legacy consumer settings just for the duration of the transition.

Using the Retro Change Log Plug-In

The retro change log plug-in enables you to configure iPlanet Directory Server 5.0 to maintain a change log that is compatible with the change log implemented in Directory Server 4.0, 4.1, 4.11, 4.12, and 4.13.

Maintaining a retro change log is important in deployments where iPlanet Directory Server 5.0 coexists with iPlanet Meta Directory. You might also need to maintain a retro change log if you have directory clients that depend on a Directory Server 4.x style change log.

To use the retro change log plug-in, your Directory Server must be configured as a supplier server in a single-master replication scenario.

When you have configured iPlanet Directory Server 5.0 to maintain a retro change log, this change log is stored in a separate database with the suffix `cn=changelog`.

The change log consists of a single level of entries. Each entry in the change log has the object class `changeLogEntry`, and can include the attributes listed in Table 8-2.

Table 8-2 Attributes of a Retro Change Log Entry

Attribute	Definition
`changeNumber`	This single-valued atttribute 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. The the case of a `modrdn` operation, the `targetDN` attribute contains the DN of the entry before it was modified or moved.
`changeType`	Specifies the type of LDAP operation. This attribute can have one of the following values: **`add`**, **`delete`**, **`modify`**, or **`modrdn`**.
`changes`	For add and modify operations, contains the changes made to the entry, in LDIF format.
`newRDN`	In the case of `modrdn` operations, specifies the new RDN of the entry.
`deleteOldRdn`	In the case of `modrdn` operations, specifies whether the old RDN was deleted.
`newSuperior`	In the case of `modrdn` operations, specifies the `newSuperior` attribute of the entry.

This section explains how to perform the following procedures:

- Enabling the Retro Change Log Plug-In
- Trimming the Retro Change Log
- Searching and Modifying the Retro Change Log

Enabling the Retro Change Log Plug-In

The retro change log plug-in configuration information is held in the `cn=Retro Changelog Plugin,cn=plugins,cn=config` entry in `dse.ldif`.

To enable the retro change log plug-in from the command line:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
cn: Retro Changelog Plugin
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

2. Use the `ldapmodify` command to import the LDIF file into the directory.

For detailed information on the `ldapmodify` command, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

3. Restart the server.

For information on restarting the server, refer to “Starting and Stopping the Directory Server,” on page 29.

The retro change log is created in the directory tree under a special suffix `cn=changelog`.

The procedure for enabling the retro change log plug-in from Directory Server Console is the same as for all Directory Server plug-ins. For information, refer to “Enabling and Disabling Plug-Ins From the Server Console,” on page 439.

Trimming the Retro Change Log

The entries in the change log can be automatically removed after a specified period of time. To configure the period of time after which entries are automatically deleted from the change log, you must set the `changelogMaximumAge` attribute in the `cn=Retro Changelog Plugin,cn=plugins,cn=config` entry.

The `changelogMaximumAge` attribute is a single-valued attribute. Its syntax is as follows:

`changelogMaximumAge:` *Integer timeUnit*

where *integer* represents a number, and *timeUnit* can be one of the following: **`s`** for seconds, **`m`** for minutes, **`h`** for hours, **`d`** for days, or **`w`** for weeks.

NOTE	There should be no space between the *Integer* and *timeUnit* variables. The space in the syntax above is intended to show that the attribute value is composed of two variable parts, not just one.

Example of `changelogMaximumAge` value:

```
changelogMaximumAge: 2d
```

Searching and Modifying the Retro Change Log

When the retro change log is created, by default, the following access control rules apply:

- 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 change log top entry `cn=changelog`.
- Write and delete access are not granted, except implicitly to the Directory Manager.

You should not grant read access to anonymous users, because the change log entries can contain modifications to sensitive information, such as passwords. Only authenticated applications and users should be allowed to access this information.

The change log supports search operations. It is optimized for searches that include filters of the form:

```
(&(changeNumber>=X)(changeNumber<=Y))
```

You should not perform add or modify operations on change log entries, although you can delete entries. However, if you want to modify the default access control policy for the change log database, you can modify the `aci` attribute of the `cn=changelog` entry.

Monitoring Replication Status

You can monitor replication status using the Directory Server Console.

To view a summary of replication status:

1. On the Directory Server Console, select the Status tab and then select Replication Status in the left navigation tree.

   In the right pane, a table appears that contains information about each of the replication agreements configured for this server.

2. Click Refresh to update the contents of the tab.

The status information displayed is described in Table 8-3.

Table 8-3 Directory Server Console - Status Tab

Table Header	Description
Agreement	Contains the name you provided when you set up the replication agreement.
Replica suffix	Contains the suffix that is replicated
Supplier	Specifies the supplier server in the agreement.
Consumer	Specifies the consumer server in the agreement.
Number of changes	Indicates the number of changes sent to this replica since the server started.
Last replica update began	Indicates when the most recent replication update started.
Last replica update ended	Indicates when the most recent replication update ended.
Last update message	Provides the status for the most recent replication updates.
Consumer initialization	Provides current status on consumer initialization (in progress or not).
Last consumer initialization update message	Provides status on the last initialization of the consumer.
Last consumer initialization began	States when the initialization of the consumer replica started.
Last consumer initialization ended	States when the initialization of the consumer replica ended.

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. Namely:

- When two entries are created on different servers with the same DN.

  The procedure for solving this type of conflict is described in "Naming Conflicts," on page 313.

- When an entry is deleted on one server, while it has child entries on the other server.

The procedure for solving this type of conflict is described in “Deleted Entries with Child Entries,” on page 315.

Entries that have a change conflict that cannot be resolved automatically by the replication process include a conflict marker attribute `nsds5ReplConflict`. The `nsdsReplConflict` attribute is an operational attribute. This makes it simple to search for entries that contain this attribute.

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

```
prompt% ldapsearch -D adminDN -w passwd -b "dc=siroe,dc=com" "nsds5ReplConflict=*"
```

For performance reasons, if you find that you have many conflicting entries every day, you may want to index the `nsds5ReplConflict` attribute. For information on indexing, refer to Chapter 10, “Managing Indexes.”

Naming Conflicts

When two entries are created with the same DN on different servers, during replication, the automatic conflict resolution procedure renames the last entry created by 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=siroe,dc=com` is created on Server A at time t1, and on Server B at time t2, and t2 is greater (or later) than t1. After replication, Server A and Server B both hold the following entries:

- `uid=adamss,ou=people,dc=siroe,dc=com` (created at time t1)
- `nsuniqueid=66446001-1dd211b2+uid=adamss,dc=siroe,dc=com` (created at time t2)

The second entry needs to be renamed in such a way that it has a unique DN. The renaming procedure depends on whether the naming attribute is single-valued or multi-valued. Each procedure is described below.

Renaming an Entry with a Multi-Valued Naming Attribute

To rename an entry that has a multi-valued naming attribute:

1. Rename the entry using a new value for the naming attribute, and keep the old RDN. For example:

```
prompt% ldapmodify -D adminDN -w passwd
```

```
>dn: nsuniqueid=66446001-1dd211b2+uid=adamss,dc=Siroe,dc=com
>changetype: modrdn
>newrdn: uid=NewValue
>deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute, and the conflict marker attribute. For example:

```
prompt% ldapmodify -D adminDN -w passwd
```

```
>dn: uid=NewValue,dc=Siroe,dc=com
>changetype: modify
>delete: uid
>uid: adamss
>-
>delete: nsds5ReplConflict
>-
```

NOTE You cannot delete the unique identifier attribute `nsuniqueid`.

Renaming an Entry with a Single-Valued Naming Attribute

To rename an entry that has a single-valued naming attribute:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:

```
prompt% ldapmodify -D adminDN -w passwd
```

```
>dn: nsuniqueid=66446001-1dd211b2+dc=pubs,dc=Siroe,dc=com
>changetype: modrdn
>newrdn: cn=TempValue
>deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute, and the conflict marker attribute. For example:

```
prompt% ldapmodify -D adminDN -w passwd
```

```
>dn: cn=TempValue,dc=Siroe,dc=com
>changetype: modify
>delete: dc
>dc: pubs
>-
>delete: nsds5ReplConflict
>-
```

NOTE You cannot delete the unique identifier attribute `nsuniqueid`.

3. Rename the entry with the intended attribute-value pair. For example:

```
prompt% ldapmodify -D adminDN -w passwd
    >dn: cn=TempValue,dc=Siroe,dc=com
    >changetype: modrdn
    >newrdn: dc=NewValue
    >deleteoldrdn: 1
```

By setting the value of the `deleteoldrdn` attribute to **`1`**, you delete the temporary attribute-value pair `cn=`*TempValue*. If you want to keep this attribute, you can set the value of the `deleteoldrdn` attribute to **`0`**.

Deleted Entries with Child Entries

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, you can either modify the glue entry to remove the `glue` object class, and the `nsds5ReplConflict` attribute, to keep the entry as a normal entry, or you can delete the glue entry and its child entries.

- The server creates a minimalistic entry with the `glue` and `extensibleObject` object classes.

  In such cases, you must modify the entry to turn it into a meaningful entry, or delete it and all of its child entries.

Controlling Access to Conflicting Entries

For reasons of interoperability with applications that rely on attribute uniqueness such as a mail server, you might need to restrict access to entries which contain the `nsds5ReplConflict` attribute.

This can be done by modifying the default ACI that grants anonymous read access, using the following command:

```
ldapmodify -h hostname -D "cn=Directory Manager" -w passwd
```

```
> dn: dc=siroe,dc=com
> changetype: modify
> delete: aci
> aci: (target ="ldap:///dc=siroe,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=siroe,dc=com")(targetattr!="userPassword")
(targetfilter="(!(nsds5ReplConflict=*))")(version 3.0;acl
"Anonymous read-search access";allow (read, search, compare)
(userdn="ldap:///anyone");)

> -
```

The new ACI contains filters out from search results all entries that contain the `nsds5ReplConflict` attribute.

Chapter 9

Extending the Directory Schema

iPlanet Directory Server comes with a standard *schema* that includes hundreds of object classes and attributes. While the standard object classes and attributes should meet most of your requirements, you may need to extend your schema by creating new object classes and attributes.

This chapter describes how to extend your schema in the following sections:

- Overview of Extending Schema
- Turning Schema Checking On and Off
- Managing Object Classes
- Managing Attributes

Overview of Extending Schema

When you add new attributes to your schema, you must create a new object class to contain them. Although it may seem convenient to just add the attributes you need to an existing object class that already contains most of the attributes you require, doing so compromises interoperability with LDAP clients.

Interoperability of Directory Server with existing LDAP clients relies on the standard LDAP schema. If you change the standard schema, you will also have difficulties when upgrading your server. For the same reasons, you cannot delete standard schema elements.

For more information on object classes, attributes, and the directory schema as well as guidelines for extending your schema, refer to *iPlanet Directory Server Deployment Guide*. For information on standard attributes and object classes, see the *iPlanet Directory Server Schema Reference*.

To extend the directory schema you should proceed in the following order:

1. Create new attributes. See "Creating Attributes," on page 320 for information.
2. Create an object class to contain the new attributes and add the attributes to the object class. See "Creating Object Classes," on page 323 for information.

Managing Attributes

Through Directory Server Console, you can view all attributes in your schema and you can create, edit, and delete your attribute extensions to the schema. The following sections describe how to manage attributes:

- "Viewing Attributes," on page 318
- "Creating Attributes," on page 320
- "Editing Attributes," on page 320
- "Deleting Attributes," on page 321

For information on managing object classes, see "Managing Object Classes," on page 321.

Viewing Attributes

To view information about all attributes that currently exist in your directory schema:

1. On the Directory Server Console, select the Configuration tab.
2. In the left navigation tree, select the Schema folder and then select the Attributes tab in the right pane.

   This tab contains information about all the standard (read-only) and user-defined attributes in the schema.

For information on the fields and lists in the Attributes tab, refer to Table 9-1.

Table 9-1 Attributes Tab Reference

Field or Pane	**Description**
Name	The name of the attribute.

Table 9-1 Attributes Tab Reference *(Continued)*

Field or Pane	**Description**
OID	The object identifier of the attribute. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If you do not specify an OID, the Directory Server automatically uses *attribute_name*`-oid`. For example, if you create the attribute `birthdate` without supplying an OID, the Directory Server automatically uses `birthdate-oid` as the OID. For more information about OIDs, or to request a prefix for your enterprise, send mail to the IANA (Internet Assigned Number Authority) at iana@iana.org or visit the IANA website at: http://www.iana.org/iana/.
Syntax	The attribute syntax: • Case Ignore String—Indicates that values for this attribute are not case sensitive. • Case Exact String—Indicates that values for this attribute are case sensitive. • Distinguished Name—Indicates that values for this attribute are DNs. • Binary—Indicates that values for this attribute are binary. • Telephone Number—Indicates that values for this attribute are in telephone number format. • Integer—Indicates that valid values for this attribute are numbers. Operational attributes are not returned as a result of an `ldapsearch` operation unless they are explicitly specified in the search. Generally, operational attributes are reserved for use by the Directory Server.
Multi	If the attribute is multivalued, an X appears in this column, otherwise, this field is blank. The Directory Server allows more than one instance of a multi-valued attribute per entry.

Creating Attributes

You can use Directory Server Console to create new attributes. After adding new attributes to your schema, you must create a new object class to contain them. See "Creating Object Classes," on page 323 for information.

To create a new attribute:

1. Display the Attributes tab.

   This procedure is explained in "Viewing Attributes," on page 318.

2. Click Create.

   The Create Attribute dialog box is displayed.

3. Enter a unique name for the attribute in the Attribute Name text box.

4. Enter an object identifier for the attribute in the Attribute OID (Optional) text box.

   OIDs are described in Table 9-1 on page 318.

5. Select a syntax that describes the data to be held by the attribute from the Syntax drop-down menu.

   Available syntaxes are described in Table 9-1 on page 318.

6. If you want the attribute to be multi-valued, select the Multi-Valued checkbox.

   The Directory Server allows more than one instance of a multi-valued attribute per entry.

7. Click OK.

Editing Attributes

You can edit only attributes you have created. You cannot edit standard attributes.

To edit an attribute:

1. Display the Attributes tab.

   This procedure is explained in "Viewing Attributes," on page 318.

2. Select the attribute that you want to edit in the User Defined Attributes table and click Edit.

   The Edit Attribute dialog box is displayed.

3. To change the attribute's name, enter a new one in the Attribute Name text box.
4. To change the attribute's object identifier, enter a new one in the Attribute OID (Optional) text box.

   OIDs are described in Table 9-1 on page 318.
5. To change the syntax that describes the data to be held by the attribute, choose a new one from the Syntax drop-down menu.
6. Available syntaxes are described in Table 9-1 on page 318.
7. To make the attribute multivalued, select the Multi-Valued checkbox.

   The Directory Server allows more than one instance of a multivalued attribute per entry.
8. When you have finished editing the attribute, click OK.

Deleting Attributes

You can delete only attributes that you have created. You cannot delete standard attributes.

To delete an attribute:

1. Display the Attributes tab.

   This procedure is explained in "Viewing Attributes," on page 318.
2. In the User Defined Attributes table, select the attribute and click Delete.
3. If prompted, confirm the delete.

   The server immediately deletes the attribute. There is no undo.

Managing Object Classes

You can use Directory Server Console to manage your schema's object classes. Through the Console, you can view all of your schema's object classes and create, edit, and delete your object class extensions to the schema. The following sections describe how to manage object classes:

- "Viewing Object Classes," on page 322

- "Creating Object Classes," on page 323
- "Editing Object Classes," on page 324
- "Deleting Object Classes," on page 325

For information on managing attributes, see "Managing Attributes," on page 318.

Viewing Object Classes

To view information about all object classes that currently exist in your directory schema:

1. On the Directory Server Console, select the Configuration tab.
2. In the navigation tree, select the Schema folder and then select the Object Classes tab in the right pane.
3. In the Object Classes list, select the object class that you want to view.

   This tab displays information about the standard or user-defined object class you selected.

For information on the fields and lists in the Object Classes tab, refer to Table 9-2.

Table 9-2 Object Classes Tab Reference

Field or Pane	Description
Parent	The parent identifies the object class from which this object class inherits its attributes and structure. For example, the parent object for the `inetOrgPerson` object class is the `organizationalPerson` object. That means that an entry with the object class `inetOrgPerson` must also include the object class `organizationalPerson`. Typically, if you want to add new attributes for user entries, the parent would be the `inetOrgPerson` object class. If you want to add new attributes for corporate entries, the parent is usually `organization` or `organizationalUnit`. If you want to add new attributes for group entries, the parent is usually `groupOfNames` or `groupOfUniqueNames`.

Table 9-2 Object Classes Tab Reference *(Continued)*

Field or Pane	Description
OID	The object identifier of the object class. An OID is a string, usually of dotted decimal numbers, that uniquely identifies an object, such as an object class or an attribute. If you do not specify an OID, the Directory Server automatically uses *ObjectClass_name*`-oid`. For example, if you create the object class `division` without supplying an OID, the Directory Server automatically uses `division-oid` as the OID. For more information about OIDs, or to request a prefix for your enterprise, send mail to the IANA (Internet Assigned Number Authority) at iana@iana.org or visit the IANA website at: `http://www.iana.org/iana/`.
Object Classes	This list contains all of the standard and user-defined object classes in the Directory Server schema.
Required Attributes	Contains a list of attributes that must be present in entries that use this object class. Includes inherited attributes.
Allowed Attributes	Contains a list of attributes that may be present in entries that use this object class. Includes inherited attributes.

Creating Object Classes

You create an object class by giving it a unique name, selecting a parent object for the new object class, and adding required and optional attributes.

To create an object class:

1. Display the Object Classes tab.

   This procedure is explained in "Viewing Object Classes," on page 322.

2. Click Create on the Object Classes tab.

   The Create Object Class dialog box is displayed.

3. Enter a unique name for the object class in the Name text box.

4. Enter an object identifier for the new object class in the OID (Optional) text box.

   OIDs are described in Table 9-2 on page 322.

5. Select a parent object for the object class from the Parent drop-down menu.

   You can choose from any existing object class. See Table 9-2 on page 322 for more information on parent object classes.

6. To add an attribute that *must* be present in entries that use the new object class, highlight the attribute in the Available Attributes list and then click the Add button to the left of the Required Attributes box.

   You can use either the standard attributes or create new ones. For information, see “Managing Attributes,” on page 318.

7. To add an attribute that may be present in entries that use the new object class, highlight the attribute in the Available Attributes list and then click the Add button to the left of the Allowed Attributes box.

8. To remove an attribute that you previously added, highlight the attribute in the Required Attributes list or the Allowed Attributes list and then click the corresponding Remove button.

   You cannot remove either allowed or required attributes that are inherited from the parent object classes.

9. When you are satisfied with your object class definition, click OK to dismiss the dialog box.

Editing Object Classes

You can use Directory Server Console to edit object classes that you previously created. You cannot edit a standard object class.

To edit an object class:

1. Display the Object Classes tab.

   This procedure is explained in “Viewing Object Classes,” on page 322.

2. Select the object class that you want to edit from the Object Classes list and click Edit.

   The Edit Object Class dialog box is displayed.

3. To change the name of the object class, enter the new name in the Name text box.

4. To change the object identifier for the object class, enter the new OID in the OID (Optional) text box.

   OIDs are described in Table 9-2 on page 322.

5. To change the parent object for the object class, select the new parent from the Parent pull-down menu.

6. To add an attribute that must be present in entries that use the new object class, highlight the attribute in the Available Attributes list and then click the Add button to the left of the Required Attributes box.

   You can either use the standard attributes or create new ones. For information, see “Managing Attributes,” on page 318.

7. To add an attribute that may be present in entries that use the new object class, highlight the attribute in the Available Attributes list and then click the Add button to the left of the Allowed Attributes box.

8. To remove an attribute that you previously added, highlight the attribute in the Required Attributes list or the Allowed Attributes list and then click the corresponding Remove button.

   You cannot remove either allowed or required inherited attributes.

9. When you are satisfied with you the object class definition, click OK to dismiss the dialog box.

Deleting Object Classes

You can delete only object classes that you have created. You cannot delete standard object classes.

To delete an object class:

1. Display the Object Classes tab.

   This procedure is explained in “Viewing Object Classes,” on page 322.

2. Select the object class that you want to remove and click Delete.

3. If prompted, confirm the delete.

   The server immediately deletes the object class. There is no undo.

Turning Schema Checking On and Off

When schema checking is on, the Directory Server ensures that:

- The object classes and attributes you are using are defined in the directory schema.
- The attributes required for an object class are contained in the entry.
- Only attributes allowed by the object class are contained in the entry.

Schema checking is turned on by default in the Directory Server, and you should always run the Directory Server with schema checking turned on. The only case where you might want to turn schema checking off is to accelerate LDAP import operations. However, there is a risk of importing entries that do not conform to the schema. Consequently, it is impossible to search for these entries.

To turn schema checking on and off:

1. On the Directory Server Console, select the Configuration tab.
2. Highlight the server icon at the top of the navigation tree, then select the Settings tab in the right pane.
3. To enable schema checking, check the “Enable Schema Checking” checkbox; clear it to turn off schema checking.
4. Click Save.

You can also turn schema checking on and off by using the `nsslapd-schemacheck` attribute. For information, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Chapter 10

Managing Indexes

The *iPlanet Directory Server Deployment Guide* guide introduced the concept of indexing, the costs and benefits and the different types of index shipped with iPlanet Directory Server. This chapter begins with a description of the searching algorithm itself, so as to place the indexing mechanism in context, and then describes how to create, delete and manage indexes. This chapter contains the following sections:

- About Indexes
- Creating Indexes
- Deleting Indexes
- Managing Indexes
- Attribute Name Quick Reference Table

About Indexes

This section provides an overview of indexing in Directory Server. It contains the following topics:

- "About Index Types," on page 328
- "About Default, System, and Standard Indexes," on page 329
- "Overview of the Searching Algorithm," on page 332
- "Balancing the Benefits of Indexing," on page 334

About Index Types

Indexes are stored in files in the directory's databases. The names of the files are based on the indexed attribute, not the type of index contained in the file. Each index file may contain multiple types of indexes if multiple indexes are maintained for the specific attribute. For example, all indexes maintained for the common name attribute are contained in the `cn.db3` file.

Directory Server supports the following types of index:

- Presence index (pres)

  The presence index contains a list of the entries that contain a particular attribute. This index is useful if, for example, you want to examine any entries that contain access control information. Generating an `aci.db3` file that includes a presence index lets you efficiently perform the search for `ACI=*` to generate the Access Control List for the server.

  The presence index is not used for base object searches.

- Equality index (eq)

  The equality index allows you to search efficiently for entries containing a specific attribute value. For example, an equality index on the `cn` attribute allows a user to perform the search for `cn=Babs Jensen` far more efficiently.

- Approximate index (approx)

  The approximate index allows efficient approximate or "sounds-like" searches. For example, an entry may include the attribute value cn=Robert E Lee. An approximate search would return this value for searches against cn~=Robert Lee, cn~=Robert, or cn~=Lee. Similarly, a search against l~=San Fransisco (note the misspelling) would return entries including l=San Francisco.

- Substring index (sub)

The substring index is a costly index to maintain, but it allows efficient searching against substrings within entries.

For example, searches of the form:

```
cn=*derson
```

would match the common names containing strings such as

```
Bill Anderson
Jill Anderson
Steve Sanderson
```

Similarly, the search for

```
telephonenumber= *555*
```

would return all the entries in your directory with telephone numbers that contain 555.

NOTE Substring indexes are limited to a minimum of two characters for each entry.

- International index

  The international index speeds up searches for information in international directories. The process for creating an international index is similar to the process for creating regular indexes, except that you apply a matching rule by associating a locale (OID) with the attributes to be indexed.

  For a listing of supported locales and their associated OIDs, refer to Appendix D, "Internationalization." If you want to configure the directory server to accept additional matching rules, contact iPlanet Professional Services.

- Browsing (virtual list view) index

  The browsing index, or virtual list view index, speeds up the display of entries in the Directory Server Console. This index is particularly useful if you branch of your directory contains hundreds of entries, for example, the `ou=people` branch. You can create a browsing index on any branchpoint in the directory tree to improve display performance. You do this through the Directory Server Console or by using the `vlvindex` command-line tool.

About Default, System, and Standard Indexes

When you install Directory Server a set of default and system indexes is created per database instance. To maintain these indexes, the directory uses standard indexes.

Overview of Default Indexes

The default indexes can be modified depending on your indexing needs, although you should ensure that no server plug-ins or other servers in your enterprise depend on this index before you remove it.

The following tables lists the default indexes installed with the directory:

Table 10-1 Default indexes

Attribute	Eq	Pres	Sub	Purpose
cn	X	X	X	Improves the performance of the most common types of user directory searches.
givenName	X	X	X	Improves the performance of the most common types of user directory searches.
mail	X	X	X	Improves the performance of the most common types of user directory searches.
mailHost	X			Used by the iPlanet Messaging Server.
member	X			Improves iPlanet server performance. This index is also used by the referential integrity plug-in. See "Maintaining Referential Integrity," on page 64 for more information.
owner	X			Improves iPlanet server performance. This index is also used by the referential integrity plug-in. See *iPlanet Directory Server Administrator's Guide* for more information.
seeAlso	X			Improves iPlanet server performance. This index is also used by the referential integrity plug-in. See "Maintaining Referential Integrity," on page 64 for more information.
sn	X	X	X	Improves the performance of the most common types of user directory searches.
telephoneNumber	X	X	X	Improves the performance of the most common types of user directory searches.
uid	X			Improves iPlanet server performance.
uniquemember	X			Improves iPlanet server performance. This index is also used by the referential integrity plug-in. See "Maintaining Referential Integrity," on page 64 for more information.

Overview of System Indexes

System indexes are indexes that cannot be deleted or modified. They are required by the directory to function properly. The following table lists the system indexes included with the directory:

Table 10-2 System indexes

Attribute	Eq	Pres	Purpose
aci		X	Allows the directory server to quickly obtain the access control information maintained in the database.
dnComp	X		Used to help accelerate subtree searches in the directory.
objectClass	X		Used to help accelerate subtree searches in the directory.
entryDN	X		Speeds up entry retrieval based on DN searches.
parentID	X		Enhances directory performance during one-level searches.
numSubordinates		X	Used by the Directory Server Console to enhance display performance on the Directory tab.
nsUniqueID	X		Used to search for specific entries.

Overview of Standard Indexes

Because of the need to maintain default indexes and other internal indexing mechanisms, the Directory Server also maintains certain standard index files. The following standard indexes exist by default. You do not need to generate them:

- `id2entry.db3`—contains the actual directory database entries. All other database files can be recreated from this one.
- `id2children.db3`—restricts the scope of one-level searches, that is, searches that examine an entry's immediate children.
- `dn.db3`—controls the scope of subtree searches; that is, searches that examine an entry and all the entries in the subtree beneath it.
- `dn2id.db3`—begins all searches efficiently by mapping an entry's distinguished name to its ID number.

Overview of the Searching Algorithm

Indexes are used to speed up searches. To understand how the directory uses indexes, it helps to understand the searching algorithm. Each index contains a list of attributes (such as the `cn`, common name, attribute) and a pointer to the entries corresponding to each value. Directory Server processes a search request as follows:

1. An LDAP client application, such as Netscape Communicator or Directory Server Gateway sends a search request to the directory.
2. The directory examines the incoming request to make sure that the specified base DN matches a suffix contained by one or more of its databases or database links.
   - If they do match, the directory processes the request.
   - If they do not match, the directory returns an error to the client indicating that the suffix does not match. If a referral has been specified in the `nsslapd-referral` attribute under `cn=config`, the directory also returns the LDAP URL where the client can attempt to pursue the request.
3. If the search request for each database attribute can be satisfied by a single index, then the server reads that index to generate a list of potential matches.

   If there is no index for the attribute, the directory generates a candidate list that includes all entries in the database which makes the search considerably slower. (The directory will also do this if the All IDs token is set for the index key that the server is using. For information on All IDs, see “Managing Indexes,” on page 353.)

   If a search request contains multiple attributes, the directory consults multiple indexes and then combines the resulting lists of candidate entries.
4. If there is an index for the attribute, the directory takes the candidate matches from the index files in the form of a series of entry ID numbers.
5. The directory uses the returned entry ID numbers to read the corresponding entries from the `id2entry.db3` file. The directory server then examines each of the candidate entries to see if any match the search criteria. The directory returns matching entries to the client as each is found.

   The directory continues until it has either examined all candidate entries, or until 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.

See *iPlanet Directory Server Configuration, Command, and File Reference* for further information about these attributes.

In addition, the directory uses a variation of the metaphone phonetic algorithm to perform searches on an approximate index. Each value is treated as a sequence of words, and a phonetic code is generated for each word.

NOTE The metaphone phonetic algorithm in Directory Server 5.0 supports only US-ASCII letters. Therefore, use approximate indexing only with English values.

Values entered on an approximate search are similarly translated into a sequence of phonetic codes. An entry is considered to match a query if both of the following are true:

- All of the query string codes match the codes generated in the entry string.
- All of the query string codes are in the same order as the entry string codes.

For example:

Name in the directory (Phonetic code)	Query string (Phonetic code)	Match comments
Alice B Sarette (ALS B SRT)	Alice Sarette (ALS SRT)	Matches. Codes are specified in the correct order.
	Alice Sarrette (ALS SRT)	Matches. Codes are specified in the correct order despite the misspelling of Sarette.

Name in the directory (Phonetic code)	Query string (Phonetic code)	Match comments
	Surette (SRT)	Matches. The generated code exists in the original name despite the misspelling of Sarette.
	Bertha Sarette (BR0 SRT)	No match. The code BR0 does not exist in the original name.
	Sarette, Alice (SRT ALS)	No match. The codes are not specified in the correct order.

Balancing the Benefits of Indexing

Before you create new indexes, balance the benefits of maintaining indexes against the costs. Keep in mind that:

- Approximate indexes are not efficient for attributes commonly containing numbers, such as telephone numbers.
- Substring indexes do not work for binary attributes. Equality indexes should be avoided if the value is big (such as attributes intended to contain photographs or passwords containing encrypted data).
- Maintaining indexes for attributes not commonly used in a search increase 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.
- Keep in mind that the more indexes you maintain, the more disk space you will require.

The following example illustrates exactly how time consuming indexes can become. Consider the procedure for creating a specific attribute:

1. The directory server receives an add or modify operation.
2. The directory server examines the indexing attributes to determine whether an index is maintained for the attribute values.
3. If the created attribute values are indexed, then the directory server generates the new index entries.
4. Once the server completes the indexing, the actual attribute values are created according to the client request.

For example, suppose the directory server is asked to add the entry

```
dn: cn=Bill Pumice, ou=People, o=siroe.com
objectclass: top
objectClass: person
objectClass: orgperson
objectClass: inetorgperson
cn: Bill Pumice
cn: Bill
sn: Pumice
ou: Manufacturing
ou: people
telephonenumber: 408 555 8834
description: Manufacturing lead for the Z238 line.
```

Further suppose that the directory server is maintaining the following indexes:

- Equality, approximate, and substring indexes for common name and surname attributes
- Equality and substring indexes for the telephone number attribute
- Substring indexes for the description attribute

Then to add this entry to the directory, the directory server must perform these steps:

1. Create the common name equality index entry for “Bill” and “Bill Pumice”.
2. Create the appropriate common name approximate index entries for “Bill” and “Bill Pumice”.
3. Create the appropriate common name substring index entries for “Bill” and “Bill Pumice”.
4. Create the surname equality index entry for “Pumice”.
5. Create the appropriate surname approximate index entry for “Pumice”.
6. Create the appropriate surname substring index entries for “Pumice”.
7. Create the telephonenumber equality index entry for “408 555 8834”.
8. Create the appropriate telephonenumber substring index entries for “408 555 8834”.
9. Create the appropriate description substring index entries for “Manufacturing lead for the Z238 line of widgets.” A large number of substring entries are generated for this string.

The example shows that indexing can be costly.

Creating Indexes

This section describes how to create presence, equality, approximate, substring and international indexes for specific attributes using the Directory Server Console and the command line.

NOTE Given that iPlanet Directory Server 5.0 can operate in either a single or multi-database environment, you need to remember to create your new indexes in every database instance, since newly created indexes are not automatically created in the other databases.

However, the same is not entirely true for default indexes because they are automatically present and maintained in subsequent database instances, but not added to existing ones. In other words, the directory uses your most recently created set of default indexes in subsequent databases. This means that if you add a default index to your second database instance, it will not be maintained in your first database instance but will be maintained in any subsequent instances.

As the procedure for creating browsing indexes is different, it is covered in a separate part.

This section contains the following procedures:

- Creating Indexes From the Server Console
- Creating Indexes From the Command Line
- Creating Browsing Indexes From the Server Console
- Creating Browsing Indexes from the Command Line

Creating Indexes From the Server Console

Using the Directory Server Console you can create presence, equality, approximate, substring, and international indexes for specific attributes.

To create indexes:

1. On the Directory Server Console, select the Configuration tab.
2. Expand the Data node, then expand the suffix of the database you want to index and select the database.
3. Select the Indexes tab in the right pane.

NOTE Do not click on the Database Settings node because this will take you to the Default Index Settings window and not the window for configuring indexes per database.

4. If the attribute you want to index is listed in the Additional Indexes table, skip to Step 6. Otherwise, click Add Attribute.

   A dialog box appears containing a list of all of the available attributes in the server schema.

5. Select the attribute you want to index and click OK.

   The server adds the attribute to the Additional Indexes table.

6. Select the checkbox for each type of index you want to maintain for each attribute.
7. If you want to create an index for a language other than English, enter the OID of the collation order you want to use in the Matching Rules field.

   You can index the attribute using multiple languages by listing multiple OIDs separated by commas (but no whitespace). For a list of languages, their associated OIDs and further information regarding collation orders see Appendix D, “Internationalization”.

8. Click Save.

   The Indexes dialog box appears displaying the status of the index creation and informing you when the indexes have been created. You can click on the Status Logs box to view the status of the indexes created. Once the indexing is complete, click Close to close the Indexes dialog box.

The new index is immediately active for any new data that you add and any existing data in your directory. You do not have to restart your server.

Creating Indexes From the Command Line

You can create presence, equality, approximate, substring and international indexes for specific attributes from the command line.

Creating indexes from the command line involves two steps:

- Using the `ldapmodify` command-line utility to add a new index entry or edit an existing index entry.
- Running the `db2index.pl` perl script to generate the new set of indexes to be maintained by the server.

NOTE	You cannot create new system indexes because system indexes are hard coded in Directory Server.

The following sections describe the steps involved in creating indexes.

Adding an Index Entry

Use `ldapmodify` to add the new index attributes to your directory. If you want to create a new index that will become one of the default indexes, add the new index attributes to the `cn=default indexes,cn=config,cn=ldbm database, cn=plugins,cn=config` entry.

To create a new index for a particular database, add it to the `cn=index,cn=`*`instanceName`*`,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=`*`instanceName`* corresponds to the name of the database.

For information on the LDIF update statements required to add entries see "LDIF Update Statements," on page 54.

For example, you want to create presence, equality and substring indexes for the `sn` (surname) attribute in the Siroe1 database.

First, type the following to change to the directory containing the utility:

```
cd /usr/iplanet/servers/shared/bin
```

Run the `ldapmodify` command-line utility as follows:

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w password
```

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

Next, you add the following entry for the new indexes:

```
dn: cn=sn,cn=index,cn=Siroe1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

The `cn` attribute contains the name of the attribute you want to index, in this example the `sn` attribute. The entry is a member of the `nsIndex` object class. The `nsSystemIndex` attribute is `false`, indicating that the index is not essential to Directory Server operations. The multi-valued `nsIndexType` attribute specifies the presence (`pres`), equality (`eq`) and substring (`sub`) indexes. Note that each keyword has to be entered on a separate line. The `nsMatchingRule` attribute specifies the OID of the Bulgarian collation order.

Specifying an index entry with no value in the `nsIndexType` attribute results in all indexes (except international) being maintained for the specified attribute. For example, suppose instead that you specify the following entry for your new `sn` indexes:

```
dn: cn=sn,cn=index,cn=instance,cn=ldbm
database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:
```

This new entry results in all indexes for the `sn` (surname) attribute.

You can use the keyword `none` in the `nsIndexType` attribute to specify that no indexes are to be maintained for the attribute. For example, suppose you want to temporarily disable the sn indexes you just created on the Siroe1 database,. You change the `nsIndexType` to `none` as follows:

```
dn: cn=sn,cn=index,cn=Siroe1,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, refer to Appendix D, "Internationalization."

For more information about the index configuration attributes, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

For more information about the `ldapmodify` command-line utility, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

NOTE You should always use the attribute's primary name (not the attribute's alias) when creating indexes. The primary name of the attribute is the first name listed for the attribute in the schema, for example `uid` for the userid attribute. See Table 10-3 on page 359 for a list of all primary and alias attribute names.

Running the db2index.pl Script

Once you have created an indexing entry or added additional index types to an existing indexing entry, run the `db2index.pl` script to generate the new set of indexes to be maintained by the Directory Server. Once you run the script, the new set of indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `db2index.pl` perl script:

1. From the command line, change to the following directory:
   `/usr/iplanet/servers/slapd-`*`serverID/`*

   where *`serverID`* is the name of your directory server.

2. Run the `db2index.pl` perl script.

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

Two examples of generating indexes using the `db2index.pl` follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl db2index.pl -D "cn=Directory Manager"
 -w password -n SiroeServer -t sn
```

CAUTION You need to run the script from the following directory on NT machines: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

```
db2index.pl -D "cn=Directory Manager" -w passsword
 -n SiroeServer -t sn
```

The following table describes the `db2index.pl` options used in the examples:

Option Name	Description
-D	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
-n	Specifies the name of the database into which you are importing the data.
-t	Specifies the name of the attribute contained by the index.

For more information about the `db2index.pl` perl script, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Creating Browsing Indexes From the Server Console

To create a browsing index using the Directory Server Console:

1. On the Directory Server Console, select the Directory tab.
2. Select the entry for which you want to create the index in the left navigation tree (for example, `People`) and select Create Browsing Index from the Object menu.

   You can also select and right-click the entry for which you want to create the index in the navigation tree and choose Create Browsing Index from the pop-up menu.
3. The Create Browsing Index dialog box appears displaying the status of the index creation. You can click on the Status Logs box to view the status of the indexes created.
4. Click Close to close the Create Browsing Index dialog box.

   The new index is immediately active for any new data that you add to your directory. You do not have to restart your server.

Creating Browsing Indexes from the Command Line

Creating a browsing index, or virtual list view (VLV) index from the command line involves two steps:

- Using the `ldapmodify` to add new browsing index entries or edit existing browsing index entries.
- Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server.

The following sections describe the steps involved in creating browsing indexes.

Adding a Browsing Index Entry

The type of browsing index entry we want to create depends on the type of `ldapsearch` attribute sorting we want to accelerate. It is important to take the following items into account:

- The scope of the search (base, one, sub). For more information on the `ldapsearch -s` option which allows you to specify the scope of searches, see the *iPlanet Directory Server Configuration, Command, and File Reference*.
- The base of the search (the entry you want to use as a starting point for the search). For more information on the `ldapsearch -b` option which allows you to specify the base of searches, see the *iPlanet Directory Server Configuration, Command, and File Reference*.
- The attributes you want to sort,
- The filter of the search. For more information on specifying filters for searches see Appendix B, “Finding Directory Entries”.
- The ldbm database to which the entry, that forms the base of the search, belongs.

NOTE	You can only create browsing indexes in ldbm databases.

For example, you want to create a browsing index to accelerate an `ldapsearch` on the entry `"dc=Siroe,dc=com"` held in the `Siroe1` database where the search base is `"dc=Siroe,dc=com"`, the search filter is `(|(objectclass=*)(objectclass=ldapsubentry))`, the scope is `one` and the sorting order for the returned attributes is `cn`, `givenname`, `o`, `ou`, and `sn`.

First, type the following to change to the directory containing the utility:

```
cd /usr/iplanet/servers/shared/bin
```

Run the `ldapmodify` command-line utility as follows:

```
ldapmodify -a -h server -p 389 -D "cn=directory manager" -w password
```

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

Next, you need to add two browsing index entries which define your browsing index.

The first entry you add specifies the base, scope and filter of the browsing index:

```
dn: cn="dc=Siroe,dc=com",cn=Siroe1,cn=ldbm
database,cn=plugins,cn=config
objectClass:top
objectClass:vlvSearch
cn:"dc=Siroe,dc=com"
vlvbase:"dc=Siroe,dc=com"
vlvscope:one
vlvfilter:(|(objectclass=*)(objectclass=ldapsubentry))
```

The `cn` contains the browsing index identifier which specifies the entry on which you want to create the browsing index, in this example the `"dc=Siroe,dc=com"` entry. We recommend you use the `dn` of the entry for your browsing index identifier, which is, the approach adopted by the Directory Server Console, to prevent identical browsing indexes from being created. The entry is a member of the `vlvSearch` object class. The `vlvbase` attribute value specifies the entry on which you want to create the browsing index, in this example the `"dc=Siroe,dc=com"` entry (that is the browsing index identifier). The `vlvscope` attribute is `one`, indicating that the base for the search you want to accelerate is `one`. A search base of `one` means that only the immediate children of the entry specified in the `cn` attribute, and not the entry itself, will be searched. The `vlvfilter` specifies the filter to be used for the search, in this example `(|(objectclass=*)(objectclass=ldapsubentry))`.

The second entry you add specifies the sorting order you want for the returned attributes:

```
dn:cn=sort_cn_givenname_o_ou_sn,cn="dc=Siroe,dc=com",cn=Siroe1,
cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:vlvIndex
cn:cn=sort_cn_givenname_o_ou_sn
vlvsort:cn givenname o ou sn
```

The `cn` contains the browsing index sort identifier. We recommend you use a sort identifier which clearly identifies the search sorting order for the browsing index you create, such as the explicit sort identifier `cn=sort_cn_givenname_o_ou_sn` in this example. 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=`*instanceName*`,cn=ldbm database,cn=plugins,cn=config` directory tree node and the second entry *must* be a child of the first entry.

Running the vlvindex Script

Once you have created the two browsing indexing entries or added additional attribute types to an existing indexing browsing entries, run the `vlvindex` script to generate the new set of browsing indexes to be maintained by the Directory Server. Once you run the script, the new set of browsing indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `vlvindex` script:

1. From the command line, change to the following directory: `/usr/iplanet/servers/slapd-`*`serverID/`*

   where *`serverID`* is the name of your directory server.

2. Run the `vlvindex` script.

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

Two examples of generating browsing indexes using the `vlvindex` script follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl vlvindex -n Siroe1
-T "dc=Siroe,dc=com"
```

CAUTION You need to run the script from the following directory on NT machines: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

```
vlvindex -n Siroe1 -T "dc=Siroe,dc=com"
```

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

Option Name	Description
-n	Name of the database containing the entries to index.
-T	Browsing index identifier to use to create browsing indexes.

For more information about the `vlvindex` script, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Deleting Indexes

This section describes how to delete presence, equality, approximate, substring, international and browsing indexes for specific attributes.

NOTE Since iPlanet Directory Server 5.0 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.

As the procedure for deleting browsing indexes is different, it is covered in a separate section. This section contains the following procedures:

- Deleting Indexes From the Server Console

- Deleting Indexes From the Command Line
- Deleting Browsing Indexes From the Server Console

CAUTION You must not delete system indexes as deleting them can significantly affect Directory Server performance. System indexes are located in the `cn=index,cn=instance,cn=ldbm database,cn=plugins,cn=config` entry and the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

Take care when deleting default indexes as this can also affect how Directory Server works.

For further information on system and default indexes see the *iPlanet Directory Server Deployment Guide*.

Deleting Indexes From the Server Console

Using the Directory Server Console you can delete indexes you have created, indexes used by other iPlanet servers (such as Messaging or Calendar server), and default indexes. You cannot delete system indexes.

To delete indexes using the Directory Server Console:

1. On the Directory Server Console, select the Configuration tab.
2. Expand the Data node and expand the suffix associated with the database containing the index. Select the database from which you want to delete the index.
3. Locate the attribute containing the index you want to delete. Clear the checkbox under the index.

   If you want to delete all indexes maintained for a particular attribute, select the attribute's cell under Attribute Name and click Delete Attribute.
4. Click Save.

   A Delete Index warning dialog box appears asking you to confirm that you want to delete the index. Click Yes to delete the index.
5. The Delete Browsing Index dialog box appears displaying the status of the index deletion. You can click on the Status Logs button to view the status of the indexes deleted. Once the indexing is complete, click on Close to close the Delete Browsing Index box.

Deleting Indexes From the Command Line

You can browsing index, or virtual list view (VLV) indexes using the `ldapdelete` command-line utility as follows:

- Delete an entire index entry or delete unwanted index types from an existing index entry using the `ldapdelete` command-line utility.
- Generate the new set of indexes to be maintained by the server using the `db2index.pl` script.

The following sections describe the steps involved in deleting an index.

Deleting an Index Entry

Use the `ldapdelete` command-line utility to delete either the entire indexing entry or the unwanted index types from an existing entry.

If you want to delete the indexes for a particular database, you remove your index entry from the `cn=index,cn=`*`instanceName`*`,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=`*`instanceName`* corresponds to the name of the database.

To delete a default index, remove it from the `cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

For example, you want to delete presence, equality and substring indexes for the sn attribute on the database named Siroe1.

You want to delete the following entry:

```
dn: cn=sn,cn=index,cn=Siroe1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

To run the `ldapdelete` command-line utility, type the following to change to the directory containing the utility:

```
cd /usr/iplanet/servers/shared/bin
```

Perform the `ldapdelete` as follows:

```
ldapdelete -D "cn=Directory Manager" -w password -h SiroeServer
-p845 "cn=sn,cn=index,cn=Siroe1,dn=ldbm
database,cn=plugins,dn=config"
```

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

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

For full information on `ldapdelete` options, refer to the *iPlanet Directory Server Configuration, Command, and File Reference.*

Once you have deleted this entry, the presence, equality, and substring indexes for the `sn` attribute will no longer be maintained by the Siroe1 database.

Running the db2index.pl Script

Once you have deleted an indexing entry or deleted some of the index types from an indexing entry, run the `db2index.pl` script to generate the new set of indexes to be maintained by the Directory Server. Once you run the script, the new set of indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `db2index.pl` perl script:

1. From the command line, change to the following directory:
   `/usr/iplanet/servers/slapd-`*`serverID/`*

   where *`serverID`* is the name of your directory server.

2. Run the `db2index.pl` perl script.

   For more information about using the `db2index.pl` perl script, refer to *iPlanet Directory Server Configuration, Command, and File Reference.*

Two examples of generating the new set of indexes to be maintained by the server using `db2index.pl` follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl db2index.pl -D "cn=Directory Manager"
 -w password -n Siroe1
```

CAUTION You need to run the script from the following directory on NT machines: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

```
db2index.pl -D "cn=Directory Manager" -w password
 -n Siro31
```

The following table describes the `db2index.pl` options used in the examples:

Option Name	Description
`-D`	Specifies the DN of the administrative user.
-w	Specifies the password of the administrative user.
`-n`	Specifies the name of the database into which you are importing the data.

For more information about the `db2index.pl` perl script, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Deleting Browsing Indexes From the Server Console

Using Directory Server Console you can delete browsing indexes.

To delete a browsing index using the Directory Server Console:

1. On the Directory Server Console, select the Database tab.
2. Select the entry from which you want to delete the index in the navigation tree, for example, `People`, and select Delete Browsing Index from the Object menu.You can also select and right-click the entry for which you want to create the index in the navigation tree and choose Delete Browsing Index from the pop-up menu.

3. A Delete Browsing Index dialog box appears asking you to confirm that you want to delete the index. Click Yes to delete.
4. The Delete Browsing Index dialog box appears displaying the status of the index deletion.

Deleting Browsing Indexes From the Command Line

Deleting a browsing index, or virtual list view (VLV) index from the command line involves two steps:

- Using the `ldapdelete` to delete browsing index entries or edit existing browsing index entries.
- Running the `vlvindex` script to generate the new set of browsing indexes to be maintained by the server.

The following sections describe the steps involved in deleting browsing indexes.

Deleting a Browsing Index Entry

Use the `ldapdelete` command-line utility to either delete browsing indexing entries or edit existing browsing index entries.

To delete browsing indexes for a particular database, you remove your browsing index entries from the `cn=index,cn=`*`instanceName`*`,cn=ldbm database,cn=plugins,cn=config` entry, where `cn=`*`instanceName`* corresponds to the name of the database.

For example, you want to delete a browsing index for accelerating `ldapsearch` operations on the entry "`dc=Siroe,dc=com`" held in the `Siroe1` database where the search base is "`dc=Siroe,dc=com`" the search filter is `(|(objectclass=*)(objectclass=ldapsubentry))`, the scope is `one` and the sorting order for the returned attributes is `cn`, `givenname`, `o`, `ou`, and `sn`.

To delete this browsing index you need to delete the two corresponding browsing index entries which follow:

```
dn: cn="dc=Siroe,dc=com",cn=Siroe1,cn=ldbm
database,cn=plugins,cn=config
objectClass:top
objectClass:vlvSearch
```

```
cn:"dc=Siroe,dc=com"
vlvbase:"dc=Siroe,dc=com
vlvscope:one
vlvfilter:(|(objectclass=*)(objectclass=ldapsubentry))
```

and

```
dn:cn=sort_cn_givenname_o_ou_sn,cn="dc=Siroe,dc=com",cn=Siroe1,
cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:vlvIndex
cn:cn=sort_cn_givenname_o_ou_sn
vlvsort:cn givenname o ou sn
```

To run the `ldapdelete` command-line utility, type the following to change to the directory containing the utility:

```
cd /usr/iplanet/servers/shared/bin
```

Perform the `ldapdelete` as follows:

```
ldapdelete -D "cn=Directory Manager" -w password -h SiroeServer -p
845 "cn="dc=Siroe,dc=com",cn=Siroe1,cn=ldbm
database,cn=plugins,cn=config"
"cn=sort_cn_givenname_o_ou_sn,cn="dc=Siroe,dc=com",cn=Siroe1,
cn=ldbm database,cn=plugins,cn=config"
```

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

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

For full information on `ldapdelete` options, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

Once you have deleted these two browsing index entries, the browsing index for accelerating `ldapsearch` operations on the entry "`dc=Siroe,dc=com`" held in the `Siroe1` database where the search base is "`dc=Siroe,dc=com`" the search filter is `(|(objectclass=*)(objectclass=ldapsubentry))`, the scope is one and the sorting order for the returned attributes is `cn`, `givenname`, `o`, `ou`, and `sn`. will no longer be maintained by the `Siroe1` database

Running the vlvindex Script

Once you have deleted browsing indexing entries or deleted unwanted attribute types from existing browsing indexing entries, run the `vlvindex` script to generate the new set of browsing indexes to be maintained by the Directory Server. Once you run the script, the new set of browsing indexes is active for any new data you add to your directory and any existing data in your directory.

To run the `vlvindex` script:

1. From the command line, change to the following directory: `/usr/iplanet/servers/slapd-`*`serverID/`*

   where *`serverID`* is the name of your directory server.

2. Run the `vlvindex` script.

   For more information about using the `vlvindex` script, refer to *iPlanet Directory Server Configuration, Command, and File Reference.*

Two examples of creating indexes using `vlvindex` follow:

Windows NT batch file:

```
..\bin\slapd\admin\bin\perl vlvindex -n Siroe1
-T "dc=Siroe,dc=com"
```

CAUTION You need to run the script from the following directory on NT machines: `..\bin\slapd\admin\bin\perl`. This path appears in the example.

UNIX shell script:

```
vlvindex -n Siroe1 -T "dc=Siroe,dc=com"
```

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

Option Name	Description
-n	Name of the database containing the entries to index.

Option Name	Description
-T	Browsing index identifier to use to create browsing indexes.

For more information about the `vlvindex` script, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Managing Indexes

Each index that the directory uses is composed of a table of index keys and matching entry ID lists. This entry ID list is used by the directory to build a list of candidate entries that may match a client application's search request (see "About Indexes," on page 327 for details).

For each entry ID list there is a size limit as specified in the `nsslapd-allidsthreshold` attribute. This size limit is globally applied to all index keys managed by the server and is logically called All IDs Threshold. When the size of an individual ID list reaches this limit, the server replaces that entry ID list with an All IDs token.

The All IDs token 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 that type of search. The directory assumes that some other aspect of the search request will allow the server to narrow its candidate list before processing the request.

The following sections examine the benefits and drawbacks of the All IDs mechanism. They also give advice for the tuning of the All IDs Threshold.

Benefits of the All IDs Mechanism

The All IDs mechanism is an important mechanism for improving search performance in those cases where the search results would be most or all directory entries (for example, searches such as `cn=*`). By assuming that all entry IDs are returned Directory Server:

- Does not have to maintain infinitely increasing entry ID lists, thus minimizing your Directory Server's disk space usage

- Does not have to load unnecessarily large entry ID lists into memory in response to search requests that result in all directory entries anyway, thus increasing search performance by reducing large disk reads
- Does not require large amounts of RAM to hold in memory unnecessarily large entry ID lists

Drawbacks of the All IDs Mechanism

Performance problems can occur if the All IDs threshold is set either too low (this is the most common problem) or too high for your directory's size.

When All IDs Threshold is Too Low

When you set the All IDs Threshold too low, too many index keys will contain the All IDs token. This can result in too many directory searches examining every entry in your directory. The performance hit on searches can be considerable.

For example, suppose you are managing an equality index on the common name (`cn`) attribute. One of the index keys stored in your `cn` index is `cn=James`. The corresponding entry ID list contains the ID number of every entry containing an attribute that is set to `James`.

The equality index on the `cn` attribute is easy to maintain because only small fraction of the entries in your directory will include `cn=James`. Performance for searches that use a `cn=James` filter will be improved because only that small fraction of entry IDs needs to be examined when servicing the search request.

However, over time your directory may continue to grow. As it does, more and more `James` may be added, but at the same relatively small proportion of total directory entries. Eventually, the `cn=James` entry ID list can become quite large, but it will still be a list that is necessary for search performance. If your directory grows large enough that so many `cn=James` entries are added that the All IDs threshold is met, then the `cn=James` entry ID list is replaced with an All IDs token. Every time you search for `cn=James`, the directory server will examine every single entry in the directory in response to the search request.

When the database grows large, the All IDs threshold will be set for a large percentage of all index keys and your search performance will significantly degrade.

When All IDs Threshold is Too High

Setting the All IDs Threshold too high can also cause performance problems. An excessively high All IDs Threshold results in large entry ID lists that must be maintained and loaded into memory when servicing search requests. An excessively high All IDs Threshold can eliminate all of the benefits of the All IDs mechanism (see “Benefits of the All IDs Mechanism,” on page 353 for details).

All IDs Threshold Tuning Advice for Single-Enterprise Directories

Be careful when changing the default All IDs Threshold value for your server. If you change the threshold to an inappropriate value, you can compromise rather than improve your server performance. This tuning advice is intended primarily for single-enterprise directories of up to 80,000 entries.

If your directory size is stable, set the All IDs Threshold to about 5 percent of the total number of entries stored in your directory. That is, if you have 50,000 entries in your directory, set the All IDs Threshold to 2,500.

If, you plan to add large numbers of entries to your directory in the near future, you should carefully consider your All IDs Threshold value. Consider the following:

- Changing the All IDs Threshold means that you have to rebuild your database. This is a potentially expensive operation, especially for directories that contain millions of entries.
- While we recommend setting the All IDs Threshold to 5 percent of your directory size, you should not see serious performance problems if your All IDs Threshold is as little as 0.5 percent of your current database size or as great as 50 percent of your current database size. However, we nevertheless recommend you aim to stay as close to the 5 percent rule as possible.

You need to balance your current directory needs against future expansion plans to avoid changing the All IDs Threshold at some later stage (which requires a database rebuild).

Suppose, for example, that your current directory is 50,000 entries in size. However, in the next few years you expect your directory to grow 1,000,000 entries. If you set your All IDs Threshold to 5 percent of 50,000 (2,500), then when your directory grows to 1,000,000 entries you will have a performance problem. 2,500 entries is too low for a database containing 1,000,000 entries because the lower limit for a 1,000,000 entry database is .5 percent of 1,000,000, i.e.5,000 entries.

If you expect your directory to grow considerably in the future, you can do one of the following:

- Set the All IDs Threshold to the current best value (2,500), and plan on rebuilding your database when your directory becomes large enough to warrant it. A database rebuild means shutting down your directory for however long the rebuild takes, or at least putting your directory into read-only mode. It also means reinitializing any consumer servers that your directory server is replicating entries to.
- Find a value that is a bit high for your current needs but that will work well for your future needs. For example, if your current directory contains 50,000 entries, try setting the All IDs Threshold to 20,000—that is 40 percent of 50,000 (which puts it within range of your current directory needs) and 2 percent of 1,000,000 (which puts it within range of your future directory needs).

The strategy you should choose depends on your directory deployment needs. Consider the cost of rebuilding your databases (and all associated consumer servers) versus potential affects on performance as your All IDs Threshold value moves away from the ideal setting of 5 percent.

NOTE It may make sense for you to have a different All IDs Threshold on a consumer server as it can be tuned to service different searches.

Also consider how quickly your directory will grow and how long it will take you to increase your directory size. If your directory takes years to grow, then plan to do a database rebuild. If in a few months your directory increases in size by an order of magnitude or greater, consider ways to set the All IDs Threshold so as to minimize the time between database rebuilds.

All IDs Threshold Tuning Advice for Service Providers and Extranets

For hosting service providers, extranet directories and directories with over 80,000 entries, tuning advice is available from *iPlanet Professional Services*.

Default All IDs Threshold Value

By default, the directory server is set to an All IDs Threshold of 4000. This value is suitable for a database of up to 80,000 entries. If you expect your databases to be larger than 80,000 entries, we recommend that you change your all IDs Threshold to a large value before populating your databases.

Symptoms of an Inappropriate All IDs Threshold Value

When your All IDs Threshold is set incorrectly, you will see poor search performance. However, poor search performance can be caused by other factors. For example:

- Your users are performing a lot of searches for which you are not maintaining an index.
- Your database cache size and entry cache size may be set incorrectly. See , “Tuning Directory Server Performance” on page 409 for more information.

Carefully examine these possibilities first before changing your All IDs Threshold value.

If you think that your server is suffering from an All IDs Threshold that is too low, look in your access log. See Chapter 12, “Monitoring Server and Database Activity.” Any searches that resulted in all entry IDs being returned will contain the `notes=U` flag. The `notes=U` flag will be returned for:

- Searches for which you are not maintaining an index
- Searches for which an ID list is not maintained because the All IDs Threshold value has been reached for that index key

To determine whether the search result belongs to a search that should have been indexed, you have to match the conn and op values from the RESULT line to a previous SRCH line in your access log file. The SRCH line will show the search filter that was used for the search request. If you have an index for the specified search filter, then the `notes=U` flag results from the All IDs Threshold value being reached for the index key. For example, the access log looks as follows:

```
[24/July/1998:15:12:20 -0800] conn=2 op=1 SRCH base="o=siroe.com"
scope=0 filter="(cn=James)”
```

```
[24/July/1998:15:12:20 -0800] conn=2 op=1 RESULT err=0 tag=101
nentries=10000 notes=U
```

The presence of the `notes=U` flag indicates that the All IDs Threshold has been reached for the `cn` attribute index.

Changing the All IDs Threshold Value

To change the All IDs Threshold value for your server:

1. Shut down your Directory Server.
2. Export all of your directory databases to LDIF using the command line.

   For more information, see Chapter 4, “Populating Directory Databases.”
3. Edit the `/usr/iplanet/servers/slapd-serverID/config/dse.ldif` file with the text editor of your choice or use the `ldapmodify` command-line utility to edit the `nsslapd-allidsthreshold` entry.
4. Locate the `nsslapd-allidsthreshold` attribute and change the value to the desired setting.
5. Initialize all your databases using `ldif2db`.

   See Chapter 4, “Populating Directory Databases.”
6. Restart your Directory Server.

After you increase your All IDs Threshold value, examine your database cache size.

Increasing your All IDs Threshold can result in larger memory requirements caused by larger entry ID lists. The increased in memory requirements will differ depending on the number and types of indexes that you are maintaining, but the requirements will never be larger than the factor by which you increased the `nsslapd-allidsthreshold` attribute value. That is, if you double the `nsslapd-allidsthreshold` attribute value, then you should not have to increase your database cache size to more than double its current value.

Increasing your database cache size by the same factor as you increased the All IDs Threshold is an extreme measure. If you have the physical memory available, try increasing your database cache size by a factor that is 25 percent of your `nsslapd-allidsthreshold` value increase. For example, if you doubled the All IDs Threshold value, increase your database cache size by 50 percent. If necessary, slowly increase your cache size until you are satisfied with your server’s performance.

Set your database cache size using the attribute `nsslapd-dbcachesize` attribute. For more information, see `nsslapd-dbcachesize` attribute in the *iPlanet Directory Server Configuration, Command, and File Reference*.

Attribute Name Quick Reference Table

The following 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.

Table 10-3 Attribute Name Quick Reference Table

Attribute Primary Name	Attribute Alias
dn	distinguishedName
cn	commonName
sn	surName
c	countryName
l	localityName
st	stateOrProvinceName
street	streetAddress
o	organization
ou	organizationalUnitName
facsimileTelephoneNumber	fax
uid	userId
mail	rfc822mailbox
mobile	mobileTelephoneNumber
pager	pagerTelephoneNumber
co	friendlyCountryName
labeledUri	labeledUri
ttl	timeToLive
dc	domainComponent
authorCn	documentAuthorCommonName

Table 10-3 Attribute Name Quick Reference Table

authorSn	documentAuthorSurname
drink	favoriteDrink

Chapter 11

Managing SSL

To provide secure communications over the network, iPlanet Directory Server includes the LDAPS communications protocol. LDAPS is the standard LDAP protocol, but it runs on top of the Secure Sockets Layer (SSL).

This chapter describes how to use SSL with your Directory Server in the following sections:

- Introduction to SSL in the Directory Server
- Obtaining and Installing Server Certificates
- Activating SSL
- Setting Security Preferences
- Using Certificate-Based Authentication
- Configuring LDAP Clients to Use SSL

Introduction to SSL in the Directory Server

You can use SSL to secure communications between LDAP clients and the Directory Server, or between Directory Servers that are bound by a replication agreement, or between a database link and a remote database. You can use SSL with simple authentication (bind DN and password), or with certificate-based authentication.

Using SSL with simple authentication guarantees confidentiality and data integrity. The benefits of using a certificate to authenticate to the Directory Server instead of a bind DN and password include:

- Improved efficiency

When you are using applications that prompt you once for your certificate database password, and then use that certificate for all subsequent bind or authentication operations, it is more efficient than continuously providing a bind DN and password.

- Improved security

  The use of certificate-based authentication is more secure than non-certificate bind operations. This is because certificate-based authentication uses public-key cryptography. As a result, bind credentials cannot be intercepted across the network.

Directory Server is capable of simultaneous SSL and non-SSL communications. This means that you do not have to choose between SSL or non-SSL communications for your Directory Server; you can use both at the same time.

NOTE If you are running Directory Server on a UNIX platform, enabling SSL will also enable support the the StartTLS extended operation. The StartTLS extended operation provides security on a regular LDAP connection.

Enabling SSL: Summary of Steps

To use LDAPS, you must do the following:

1. Obtain and install a certificate for your Directory Server, and configure the Directory Server to trust the certification authority's certificate.

   For information, see "Obtaining and Installing Server Certificates," on page 363.

2. Turn on SSL in your directory.

   For information, see "Activating SSL," on page 368.

3. Configure the administration server to connect to an SSL-enabled Directory Server.

   For information, see *Managing Servers with iPlanet Console*.

4. Optionally, ensure that each user of the Directory Server obtains and installs a personal certificate for all clients that will authenticate with SSL.

   For information, see "Configuring LDAP Clients to Use SSL," on page 373.

For a complete description of SSL, internet security, and certificates, see *Managing Servers with iPlanet Console*.

Obtaining and Installing Server Certificates

This section describes the process of creating a certificate database, obtaining and installing a certificate for use with your Directory Server, and configuring Directory Server to trust the certification authority's (CA) certificate.

This process is a necessary first step before you can turn on SSL in your directory. If you have already completed these tasks, see "Activating SSL," on page 368.

Obtaining and installing certificates consists of the following steps:

- Step 1: Generate a Certificate Request
- Step 2: Send the Certificate Request to the Certificate Authority
- Step 3: Install the Certificate
- Step 4: Trust the Certificate Authority
- Step 5: Confirm That Your New Certificates Are Installed

You will use the Certificate Request Wizard to generate a certificate request (Step 1) and send it to a Certificate Authority (Step 2). You then use the Certificate Install Wizard to install the certificate (Step 3), and to trust the Certificate Authority's certificate (Step 4).

These wizards automate the process of creating a certificate database, and of installing the key-pair.

Step 1: Generate a Certificate Request

To generate a certificate request and send it to a CA:

1. On the Directory Server Console, select the Tasks tab and click Manage Certificates.

   The Manage Certificates window is displayed.

2. Select the Server Certs tab, and click the Request button.

   The Certificate Request Wizard is displayed.

3. Click Next.

4. Enter the Requestor Information in the blank text fields, then click Next.

   Enter the following information:

   Server Name. Enter the fully qualified hostname of the Directory Server as it is used in DNS lookups, for example, `dir.siroe.com`.

   Organization. Enter the legal name of your company or institution. Most CAs require you to verify this information with legal documents such as a copy of a business license.

   Organizational Unit. (Optional). Enter a descriptive name for your organization within your company.

   Locality. (Optional). Enter your company's city name.

   State or Province. Enter the full name of your company's state or province (no abbreviations).

   Country. Select the two-character abbreviation for your country's name (ISO format). The country code for the United States is US. The *iPlanet Directory Server Schema Reference* contains a complete list of ISO Country Codes.

5. Enter the password that will be used to protect the private key, and click Next.

   The Next field is greyed out until you supply a password. When you click Next, the Request Submission dialog box is displayed.

6. Select Copy to Clipboard or Save to File to save the certificate request information that you must send to the Certificate Authority.

7. Click Done to dismiss the Certificate Request Wizard.

Once you have generated the request, you are ready to send it to the CA.

Step 2: Send the Certificate Request

Follow these steps to send the certificate information to the CA:

1. Use your email program to create a new email message.

2. Copy the certificate request information from the clipboard or the saved file into the body of the message.

   The content will look similar to the following example:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCVXMxEzARBgNVBAgTCkNBTElGT1JOSUExLD
AqBgVBAoTI25ldHNjYXBlIGNvbW11bmljYXRpb25zIGNvcnBvcmF0aW9uMRwwGgYDV
QQDExNtZWxsb24ubmV0c2NhcGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQK
BgQCwAbskGh6SKYOgHy+UCSLnm3ok3X3u83Us7ug0EfgSLR0f+K41eNqqWRftGR83e
mqPLDOf0ZLTLjVGJaH4Jn4l1gG+JDf/n/zMyahxtV7+mT8GOFFigFfuxJaxMjr2j7I
vELlxQ4IfZgWwqCm4qQecv3G+N9YdbjveMVXW0v4XwIDAQABoAAwDQYJKoZIhvcNAQ
EEBQADgYEAZyZAm8UmP9PQYwNy4Pmypk79t2nvzKbwKVb97G+MT/gw1pLRsI1uBoKi
nMfLgKp1Q38K5Py2VGW1E47K7/rhm3yVQrIiwV+Z8Lcc=
-----END NEW CERTIFICATE REQUEST-----
```

3. Send the email message to the CA.

Once you have emailed your request, you must wait for the CA to respond with your certificate. Response time for your request varies. For example, if your CA is internal to your company, it may only take a day or two to respond to your request. If your selected CA is external to your company, it could take several weeks to respond to your request.

When the CA sends a response, be sure to save the information in a text file. You will need the data when you install the certificate.

You should also back up the certificate data in a safe location. If your system ever loses the certificate data, you can reinstall the certificate using your backup file.

Once you receive your certificate, you are ready to install it in your server's certificate database.

Step 3: Install the Certificate

To install a server certificate:

1. On the Directory Server Console, select the Tasks tab and click Manage Certificates.

   The Manage Certificates window is displayed.

2. Select the Server Certs tab, and click Install.

   The Certificate Install Wizard is displayed.

3. Choose one of the following options for the certificate location, then click Next.

   In this file. Enter the absolute path to the certificate in this field.

   In the following encoded text block. Copy the text from the CAs email or from the text file you created and paste it in this field. For example:

```
-----BEGIN CERTIFICATE-----
MIICMjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMx
IzAhBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0wGwYDVQQLExRX
aWRnZXQgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVz
dCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3WhcNOTgwMzI2MDIzMzU3WjBP
MQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTmV0c2NhcGUgRGlyZWN0b3J5IFB1Ymxp
Y2F0aW9uczEWMBQGA1UEAxMNZHVgh49dq2itLmNvbTBaMA0GCSqGSIb3DQEBAQUA
A0kAMEYCQQCksMR/aLGdfp4m0OiGcgijG5KgOsyRNvwGYW7kfW+8mmijDtZRjYNj
jcgpF3VnlsbxbclX9LVjjNLC57u37XZdAgEDozYwNDARBglghkgBhvhCAQEEBAMC
APAwHwYDVR0jBBgwFoAU67URjwCaGqZuUpSpdLxlzweJKiMwDQYJKoZIhvcNAQEF
BQADgYEAJ+BVem3vBOP/BveNdLGfjlb9hucgmaMcQa98A/db8qimKT/ue9UGOJqL
bwbMKBBopsD56p2yV3PLJIsBgrcuSoBCuFFnxBnqSiTS/7YiYgCWqWaUAExJFmD6
6hBLseqkSWulk+hXHN7L/NrViO+7zNtKcaZLlFPf7d7j2MgX4Bo=
-----END CERTIFICATE-----
```

4. Check that the certificate information displayed is correct, and click Next.
5. Specify a name for the certificate, and click Next.
6. Verify the certificate by providing the password that protects the private key.

   This password is the same as the one you provided in “Step 1: Generate a Certificate Request,” on page 363.

Now that you have installed your certificate, you need to configure your server to trust the Certificate Authority from which you obtained the server’s certificate.

Step 4: Trust the Certificate Authority

Configuring your Directory Server to trust the certificate authority consists of obtaining your CA’s certificate and installing it into your server’s certificate database. This process differs depending on the certificate authority you use. Some commercial CAs provide a website that allows you to automatically download the certificate. Others will email it to you upon request.

Once you have the CA certificate, you can use the Certificate Install Wizard to configure the Directory Server to trust the Certificate Authority.

1. On the Directory Server Console, select the Tasks tab and click Manage Certificates.

   The Manage Certificates window is displayed.
2. Go to the CA Certs tab, and click Install.

   The Certificate Install Wizard is displayed.

3. If you saved the CA's certificate to a file, enter the path in the field provided. If you received the CA's certificate via email, copy and paste the certificate including the headers into the text field provided. Click Next.
4. Check that the certificate information that is displayed is correct, and click Next.
5. Specify a name for the certificate, and click Next.
6. Select the purpose of trusting this Certificate Authority (you can select both):

   Accepting connections from clients (Client Authentication). The server checks that the client's certificate has been issued by a trusted Certificate Authority.

   Accepting connections to other servers (Server Authentication). This server checks that the directory to which it is making a connection (for example, for replication updates) has a certificate that has been issued by a trusted Certificate Authority.
7. Click Done to dismiss the wizard.

Once you have installed your certificate and trusted the CA's certificate, you are ready to activate SSL. However, you should first make sure that the certificates have been installed correctly.

Step 5: Confirm That Your New Certificates Are Installed

1. On the Directory Server Console, select the Tasks tab and click Manage Certificates.

   The Manage Certificates window is displayed.
2. Select the Server Certs tab.

   A list of all the installed certificates for the server is displayed.
3. Scroll through the list. You should find the certificates you installed.

   Your server is now ready for SSL activation.

Activating SSL

Most of the time, you want your server to run with SSL enabled. If you temporarily disable SSL, make sure you re-enable it before processing transactions that require confidentiality, authentication, or data integrity.

Before you can activate SSL, you must create a certificate database, obtain and install a server certificate and trust the CA's certificate as described in "Obtaining and Installing Server Certificates," on page 363.

To activate SSL communications:

1. Set the secure port you want the server to use for SSL communications. See "Changing Directory Server Port Numbers," on page 31 for information.

   The encrypted port number that you specify must not be the same port number you use for normal LDAP communications. By default, the standard port number is 389 and the secure port is 636.

2. On 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.

   The tab displays the current server encryption settings.

4. Indicate that you want encryption enabled by selecting the "Enable SSL for this Server" checkbox.

5. Check the "Use this Cipher Family" checkbox.

6. Select the certificate that you want to use from the drop-down menu.

7. Click Cipher Settings .

   The Cipher Preference dialog box is displayed.

8. Select the checkbox next to the cipher you want to use, and click OK to dismiss the Cipher Preference dialog box.

   For more information about specific ciphers, see "Setting Security Preferences," on page 369.

9. Set your preferences for client authentication.

Do not allow client authentication. With this option, the server will ignore the client's certificate. This does not mean that the bind will fail.

Allow client authentication. This is the default setting. With this option, authentication is performed on the client's request. For more information about certificate-based authentication, see ""Using Certificate-Based Authentication," on page 371.

Require client authentication. With this option, the server requests authentication from the client.

NOTE If you are using certificate-based authentication with replication, then you must configure the consumer server to either allow or require client authentication.

---

10. If you want iPlanet Console to use SSL during communications with Directory Server, select Use SSL in iPlanet Console.

11. Click Save.

12. Restart the Directory Server.

See "Starting the Server with SSL Enabled," on page 33 for more information.

Setting Security Preferences

You can choose the type of ciphers you want to use for SSL communications. A *cipher* is the algorithm used in encryption. Some ciphers are more secure or *stronger* than others. Generally speaking, the more bits a cipher uses during encryption, the more difficult it is to decrypt the key. For a more complete discussion of algorithms and their strength, see *Managing Servers with iPlanet Console*.

When a client initiates an SSL connection with a server, the client tells the server what ciphers it prefers to use to encrypt information. In any two-way encryption process, both parties must use the same ciphers. There are a number of ciphers available. Your server needs to be able to use the ciphers that will be used by client applications connecting to the server.

iPlanet Directory Server provides the following SSL 3.0 ciphers:

- RC4 cipher with 40-bit encryption and MD5 message authentication.

- RC2 cipher with 40-bit encryption and MD5 message authentication.
- No encryption, only MD5 message authentication.
- DES with 56-bit encryption and SHA message authentication.
- RC4 cipher with 128-bit encryption and MD5 message authentication.
- Triple DES with 168-bit encryption and SHA message authentication.
- FIPS DES with 56-bit encryption and SHA message authentication. This cipher meets the FIPS 140-1 U.S. government standard for implementations of cryptographic modules.
- FIPS Triple DES with 168-bit encryption and SHA message authentication. This cipher meets the FIPS 140-1 US government standard for implementations of cryptographic modules.

To select the ciphers you want the server to use:

1. Make sure SSL is enabled for your server.

   For information, see "Activating SSL," on page 368.

2. On the Directory Server Console, select the Configuration tab and then select the topmost entry in the navigation tree in the left pane.
3. Select the Encryption tab in the right pane.

   This displays the current server encryption settings.

4. Click Cipher Settings.

   The Cipher Preference dialog box is displayed.

5. In theCipher Preference dialog box, specify which ciphers you want your server to use by selecting them from the list, and click OK.

   Unless you have a security reason to not use a specific cipher, you should select all of the ciphers, except for `none,MD5`.

6. On the Encryption tab, click Save.

CAUTION Avoid selecting the `none,MD5` cipher because the server will use this option if no other ciphers are available on the client. It is not secure because encryption doesn't occur.

In order to continue using the iPlanet Console with SSL, you must select at least one of the following ciphers:

- RC4 cipher with 40-bit encryption and MD5 message authentication.
- No encryption, only MD5 message authentication.
- DES with 56-bit encryption and SHA message authentication.
- RC4 cipher with 128-bit encryption and MD5 message authentication.
- Triple DES with 168-bit encryption and SHA message authentication.

Using Certificate-Based Authentication

Directory Server allows you to use certificate-based authentication for the command-line tools (which are LDAP clients) and for replication communications. Certificate-based authentication can occur between:

- An LDAP client connecting to the Directory Server
- A Directory Server connecting to another Directory Server (replication or chaining)

Setting up Certificate-Based Authentication

To set up certificate-based authentication, you must:

1. Create a certificate database for the client and the server, or for both servers involved in replication.

   On the Directory Server, the certificate database creation automatically takes place when you install a certificate. For information on creating a certificate database for a client, see “Configuring LDAP Clients to Use SSL,” on page 373.

2. Obtain and install a certificate on both the client and the server, or on both servers involved in replication.
3. Enable SSL on the server, or on both servers involved in replication.

   For information on enabling SSL, refer to “Activating SSL,” on page 368.

NOTE If iPlanet Console connects to Directory Server over SSL, selecting "Require client authentication" disables communication. This is because although iPlanet Console supports SSL, it does not have a certificate to use for client authentication.

4. Map the certificate's distinguished name to a distinguished name known by your directory.

   This allows you to set access control for the client when it binds using this certificate. This mapping process is described in *Managing Servers with iPlanet Console*.

Allowing/Requiring Client Authentication

If you have configured iPlanet Console to connect to your Directory Server using SSL *and* your Directory Server *requires* client authentication, you can no longer use iPlanet Console to manage any of your iPlanet servers. You will have to use the appropriate command-line utilities instead.

However, if at a later date you wish to change your directory configuration to no longer *require* but *allow* client authentication, so that you can use iPlanet Console, you must follow these steps:

1. Stop Directory Server.

   For information on stopping and starting the server from the command line, see "Starting/Stopping the Server From the Command Line," on page 30.

2. Modify the `cn=encryption,cn=config` entry by changing the value of the nsSSLClientAuth attribute from **`required`** to **`allowed`**.

   For information on modifying entries from the command line, see Chapter 2, "Creating Directory Entries."

3. Start Directory Server.

   You can now start iPlanet Console.

Configuring LDAP Clients to Use SSL

If you want all the users of your Directory Server to use SSL or certificate-based authentication when they connect using LDAP client applications, you must make sure they perform the following tasks:

- Create a certificate database.
- Trust the Certificate Authority (CA) that issues the server certificate.

These operations are sufficient if you want to ensure that LDAP clients recognize the server's certificate. However, if you also want LDAP clients to use their own certificate to authenticate to the directory, make sure that all your directory users obtain and install a personal certificate.

NOTE Some client applications do not verify that the server has a trusted certificate.

The following procedure describes how to use Netscape Communicator 4.7 to perform these tasks.

1. To create a certificate, it is sufficient to start Netscape Communicator 4.7.

   If it does not already exist, the certificate database will be created.

2. Use Communicator to connect to your Certificate Authority.

   If you are using an internally deployed iPlanet Certificate Server, you will go to a URL of the form:

   `https://`*hostname*`:444`

   Some Certificate Authorities provide a link that allows you to download the CA's certificate.

3. Trust the Certificate Authority.

   This task differs depending on the CA. In some cases, such as if you are connecting to a iPlanet Certificate Server, Communicator will automatically prompt you to see if you want to trust the CA.

These steps are sufficient to ensure that your client applications will accept connections to take place with the Directory Server, because the clients recognize that the Directory Server's certificate has been issued by a trusted CA.

However, if you also want the Directory Server to authenticate clients using the clients' certificate, you must perform the following additional steps:

4. On the client system, obtain a client certificate from the CA.

5. On your client system, install your client certificate.

   Regardless of how you receive your certificate (either in email or on a web page), there should be a link that you click to install the certificate. Click it and step through the dialog boxes that Communicator presents to you.

   Make sure you record the certificate information that is sent to you in a file. In particular, you must know the subject DN of the certificate because you must configure the server to map it to an entry in the directory. Your client certificate will be similar to:

```
-----BEGIN CERTIFICATE-----
MIICMjCCAZugAwIBAgICCEEwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMx
IzAhBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0wGwYDVQQLExRX
aWRnZXQgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVz
dCBUZXN0IFRlc3QgQ0EwHhcNOTgwMzEyMDIzMzU3WhcNOTgwMzI2MDIzMzU3WjBP
MQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTmV0c2NhcGUgRGlyZWN0b3J5IFB1Ymxp
Y2F0aW9uczEWMBQGA1UEAxMNZHVgh49dq2itLmNvbTBaMA0GCSqGSIb3DQEBAQUA
A0kAMEYCQQCksMR/aLGdfp4m0OiGcgijG5KgOsyRNvwGYW7kfW+8mmijDtZRjYNj
jcgpF3VnlsbxbclX9LVjjNLC57u37XZdAgEDozYwNDARBglghkgBhvhCAQEEBAMC
APAwHwYDVR0jBBgwFoAU67URjwCaGqZuUpSpdLxlzweJKiMwDQYJKoZIhvcNAQEF
BQADgYEAJ+BVem3vBOP/BveNdLGfjlb9hucgmaMcQa98A/db8qimKT/ue9UGOJqL
bwbMKBBopsD56p2yV3PLJIsBgrcuSoBCuFFnxBnqSiTS/7YiYgCWqWaUAExJFmD6
6hBLseqkSWulk+hXHN7L/NrViO+7zNtKcaZLlFPf7d7j2MgX4Bo=
-----END CERTIFICATE-----
```

6. You must convert the client certificate into binary format using the `certutil` utility. To do this:

   a. Download the `certutil` utility from http://www.iplanet.com.

      On the iPlanet home page, search for **`certutil`**. Download the most recent PKCS package. It will contain the `certutil` utility.

   b. Run `certutil` as follows:

      `certutil -L -d` *cert7.db_path* `-n` *user_cert_name* `-r >` *user_cert.bin*

      where *cert7.db_path* is the location of your certificate database, *user_cert_name* is the name you gave to your certificate when you installed it, and *user_cert.bin* is the name you must specify for the output file that will contain the binary certificate.

7. On the server, map the subject DN of the certificate that you obtained to the appropriate directory entry by editing the `certmap.conf` file.

   This procedure is described in *Managing Servers with iPlanet Console*. Make sure that the `verifyCert` parameter is set to **on** in the `certmap.conf` file.

NOTE Note that if this parameter is not set to **on**, Directory Server simply searches for an entry in the directory that matches the information in the `certmap.conf` file. If the search is successful, it grants access without actually checking the value of the `userCertificate` attribute.

8. On the Directory Server, you must 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 Editory to add the `userCertificate` attribute, with the `binary` subtype.

      When you add this attribute, instead of an editable field, the server provides a Set Value button.

   c. Click Set Value.

      A file selector is displayed. Use it to select the binary file you created in Step 6.

   For information on using the Directory Server Console to edit entries, refer to "Modifying Directory Entries," on page 41.

You can now use SSL with your LDAP clients. For information on how to use SSL with `ldapmodify`, `ldapdelete` and `ldapsearch`, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Chapter 12

Monitoring Server and Database Activity

This chapter describes monitoring database and server logs. This chapter contains the following sections:

- Viewing and Configuring Log Files
- Manual Log File Rotation
- Monitoring Server Activity
- Monitoring Database Activity
- Monitoring Database Link Activity

For information on using SNMP to monitor your server, see Chapter 13, "Monitoring Directory Server Using SNMP."

Viewing and Configuring Log Files

iPlanet Directory Server provides three types of logs to help you better manage your directory and tune performance. These logs include:

- Access Log
- Error Log
- Audit Log

The following aspects are common to the configuration of all types of logs:

- Defining a log file creation policy
- Defining a log file deletion policy

The following sections describe how to define your log file creation and deletion policy, and how to view and configure each type of log.

Defining a Log File Rotation Policy

If you want the directory to periodically archive the current log and start a new one, you can define a log file rotation policy from Directory Server Console. You can configure the following parameters:

- The total number of logs you want the directory to keep. When the directory reaches this number of logs, it deletes the oldest log file in the folder before creating a new log. The default is `10` logs. Do not set this value to `1`. If you do, the directory will not rotate the log and the log will grow indefinitely.
- The maximum size (in MB) for each log file. If you don't want to set a maximum size, type `-1` in this field. The default is `100` MB. Once a log file reaches this maximum size (or the maximum age defined in the next step), the directory archives the file and starts a new one. If you set the maximum number of logs to `1`, the directory ignores this attribute.
- How often the directory archives the current log file and creates a new one by entering a number of minutes, hours, days, weeks, or months. The default is every day. If you set the maximum number of logs to `1`, the directory ignores this attribute.

Defining a Log File Deletion Policy

If you want the directory to automatically delete old archived logs, you can define a log file deletion policy from Directory Server Console.

NOTE	The log deletion policy only makes sense if you have previously defined a log file rotation policy. Log file deletion will not work if you have just one log file. The server evaluates the log file deletion policy at the time of log rotation.

You can configure the following parameters:

- The maximum size of the combined archived logs. When the maximum size is reached, the oldest archived log is automatically deleted. If you don't want to set a maximum size, type `-1` in this field. The default is `500` MB. This parameter is ignored in the number of log files is set to `1`.
- The minimum amount of free disk space. When the free disk space reaches this minimum value, the oldest archived log is automatically deleted. The default is `5` MB. This parameter is ignored in the number of log files is set to `1`.
- The maximum age of log files. When a log file reaches this maximum age, it is automatically deleted. The default is `1` month. This parameter is ignored in the number of log files is set to `1`.

Access Log

The access log contains detailed information about client connections to the directory.

This section contains the following procedures:

- "Viewing the Access Log," on page 379
- "Configuring the Access Log," on page 380

Viewing the Access Log

To view the access log:

1. On the Directory Server Console, select the Status tab, then in the navigation tree, expand the Logs folder and select the Access Log icon.

   A table displays a list of the last 25 entries in the access log.
2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.
3. To view an archived access log, select it from the Select Log pull-down menu.
4. To display a different number of messages, enter the number you want to view in the "Lines to show" text box and then click Refresh.
5. You can display messages containing a string you specify. To do this, enter the string in the "Show only lines containing" text box and then click Refresh.

Configuring the Access Log

You can configure a number of settings to customize the access log, including where the directory stores the access log and the creation and deletion policies.

You can also disable access logging for the directory. You may do this because the access log can grow very quickly (every 2,000 accesses to your directory will increase your access log by approximately 1 MB). However, before you turn off access logging, consider that the access log provides beneficial troubleshooting information.

To configure the access log for your directory:

1. On the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder and select the Access Log icon.

   The access log configuration attributes are displayed in the right pane.

2. To enable access logging, select the Enable Logging checkbox.

   Clear this checkbox if you do not want the directory to maintain an access log.

   Access logging is enabled by default.

3. Enter the full path and filename you want the directory to use for the access log in the Log File field.

   The default is `/usr/iplanet/servers/slapd-`*serverID*`/logs/access`

4. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see "Defining a Log File Rotation Policy," on page 378.

5. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see "Defining a Log File Deletion Policy," on page 378.

6. When you have finished making changes, click Save.

Error Log

The error log contains detailed messages of errors and events the directory experiences during normal operations.

This section contains the following procedures:

- “Viewing the Error Log,” on page 381
- “Configuring the Error Log,” on page 381

Viewing the Error Log

To view the error log:

1. On the Directory Server Console, select the Status tab, then in the navigation tree, expand the Logs folder and select the Error Log icon.

   A table displays a list of the last 25 entries in the error log.

2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.
3. To view an archived error log, select it from the Select Log pull-down menu.
4. To specify a different number of messages, enter the number you want to view in the “Lines to show” text box and click Refresh.
5. You can display messages containing a string you specify. To do this, enter the string in the “Show only lines containing” text box and click Refresh.

Configuring the Error Log

You can change several settings for the error log, including where the directory stores the log and what you want the directory to include in the log.

To configure the error log:

1. On the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder and select the Error Log icon.

   The error log configuration attributes are displayed in the right pane.

2. Select the Error Log tab in the right pane.
3. To enable error logging, select the Enable Logging checkbox.

   Clear this checkbox if you do not want the directory to maintain an error log.

   Error logging is enabled by default.

4. Enter the full path and filename you want the directory to use for the error log in the Log File field.

   The default is `/usr/iplanet/servers/slapd-`*serverID*`/logs/error`

5. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see “Defining a Log File Rotation Policy,” on page 378.

6. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see “Defining a Log File Deletion Policy,” on page 378.

7. If you want to set the log level, Ctrl+click the options you want the directory to include in the Log Level list box.

   For more information about log level options, see “Log Level” in the *iPlanet Directory Server Configuration, Command, and File Reference*.

   Changing these values from the defaults may cause your error log to grow very rapidly, so it is recommended that you do not change your logging level unless you are asked to do so by iPlanet Customer Support.

8. When you have finished making changes, click Save.

Audit Log

The audit log contains detailed information about changes made to each database as well as to server configuration.

This section contains the following procedures:

- “Viewing the Audit Log,” on page 382
- “Configuring the Audit Log,” on page 383

Viewing the Audit Log

Before you can view the audit log, you must enable audit logging for the directory. See “Configuring the Audit Log,” on page 383 for information.

To view the audit log:

1. On the Directory Server Console, select the Status tab. Then, in the navigation tree, expand the Logs folder and select the Audit Log icon.

   A table displays a list of the last 25 entries in the audit log.

2. To refresh the current display, click Refresh. Select the Continuous checkbox if you want the display to refresh automatically every ten seconds.

3. To view an archived audit log, select it from the Select Log pull-down menu.
4. To display a different number of messages, enter the number you want to view in the "Lines to show" text box and click Refresh.
5. You can display messages containing a string you specify. To do this, enter the string in the "Show only lines containing" text box and click Refresh.

Configuring the Audit Log

You can use the Directory Server Console to enable and disable audit logging and to specify where the audit log file is stored.

To configure audit logging:

1. On the Directory Server Console, select the Configuration tab. Then, in the navigation tree, expand the Logs folder and select the Audit Log icon.

   The audit log configuration attributes are displayed in the right pane.

2. To enable audit logging, select the Enable Logging checkbox.

   To disable audit logging, clear the checkbox. By default, audit logging is disabled.

3. Enter the full path and filename you want the directory to use for the audit log in the field provided.

   The default is `/usr/iplanet/servers/slapd-`*serverID*`/logs/audit`

4. Set the maximum number of logs, log size, and periodicity of archiving.

   For information on these parameters, see "Defining a Log File Rotation Policy," on page 378.

5. Set the maximum size of combined archived logs, minimum amount of free disk space, and maximum age for a log file.

   For information on these parameters, see "Defining a Log File Deletion Policy," on page 378.

6. When you have finished making changes, click Save.

Manual Log File Rotation

The directory server supports automatic log file rotation for all three logs. However, you can manually rotate log files if you have not set automatic log file creation or deletion policies. By default, access, error, and audit log files can be found in the following location:

`/usr/iplanet/servers/slapd-`*serverID*`/logs/`

To manually rotate log files:

1. Shut down the server.

   See "Starting and Stopping the Directory Server," on page 29 for instructions.

2. Move or rename the log file you are rotating in case you need the old log file for future reference.

3. Restart the server.

   See "Starting and Stopping the Directory Server," on page 29 for instructions.

Monitoring Server Activity

You can monitor your directory server's current activities from either the Directory Server Console or the command line. You can also monitor the activity of the caches for all of your database. This section contains the following information:

- "Monitoring Your Server From the Directory Server Console," on page 384
- "Monitoring Your Server From the Command Line," on page 389

Monitoring Your Server From the Directory Server Console

This section contains information about using the Directory Server Console to monitor your server and the information available to you in the performance monitor.

Viewing the Server Performance Monitor

To monitor your server's activities using Directory Server Console:

1. On the Directory Server Console, select the Status tab. In the navigation tree, select Performance Counters.

   The Status tab in the right pane displays current information about server activity. If the server is currently not running, this tab will not provide performance monitoring information.

2. Click Refresh to refresh the current display. If you want the server to continuously update the displayed information, select the Continuous checkbox.

Overview of Server Performance Monitor Information

The server provides monitoring information as described in the following sections:

- General Information (Server)
- Resource Summary
- Current Resource Usage
- Connection Status
- Global Database Cache Information

General Information (Server)

The server provides the following general information:

- Server version

  Identifies the current server version.

- Configuration DN

  Identifies the distinguished name that you must use as a search base to obtain these results using the `ldapsearch` command-line utility. This field should read `cn=monitor`.

- Data version

  Provides identification information for the server's data. Usually the information shown here is only relevant if your server supplies replicas to consumer servers. The data version information is supplied as follows:

  - Server host name.

  - ❍ Server port number.
  - ❍ Database generation number. Possibly obsolete: A unique identifier that is created only when you create your directory database without a machine data entry in the LDIF file.
  - ❍ Current change log number. This is the number corresponding to the last change made to your directory. This number starts at one and increments by one for each change made to the database.

- Startup time on server

  Date and time the server was started.

- Current time on server

  Displays the current date and time on the server.

Resource Summary

The Resource Summary table displayed by the console provides the following resource-specific information:

Table 12-1 Server Performance Monitoring - Resource Summary Table

Resource	**Usage since startup**	**Average per minute**
Connections	Total number of connections to this server since server startup.	Average number of connections per minute since server startup.
Operations Initiated	Total number of operations initiated since server startup. Operations include any client requests for server action, such as searches, adds, and modifies. Often, multiple operations are initiated for each connection.	Average number of operations per minute since server startup.
Operations Completed	Total number of operations completed by the server since server startup.	Average number of operations per minute since server startup.
Entries sent to clients	Total number of entries sent to clients since server startup. Entries are sent to clients as the result of search requests.	Average number of entries sent to clients per minute since server startup.
Bytes sent to clients	Total number of bytes sent to clients since server startup.	Average number of bytes sent to clients per minute since server startup.

Current Resource Usage

The Resource Summary table in Directory Server Console provides the following resource-specific information

Table 12-2 Server Performance Monitoring - Current Resource Usage Table

Resource	Current total
Active Threads	Current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
Open Connections	Total number of open connections. Each connection can account for multiple operations, and therefore multiple threads.
Remaining Available Connections	Total number of remaining connections that the server can concurrently open. This number is based on the number of currently open connections and the total number of concurrent connections that the server is allowed to open. In most cases, the latter value is determined by the operating system, and is expressed as the number of file descriptors available to a task. On Windows NT and IBM AIX, the number of allowed concurrent connections is generated by the operating system, but is not based on file descriptors. Refer to your operating system documentation for more information.
Threads Waiting to Write to Client	Total number of threads waiting to write to the client. Threads may not be immediately written when the server must pause while sending data to a client. Reasons for a pause include a slow network, a slow client, or an extremely large amount of information being sent to the client.
Threads Waiting to Read from Client	Total number of threads waiting to read from the client. Threads may not be immediately read if the server starts to receive a request from the client and then the transmission of that request is halted for some reason. Generally, threads waiting to read are an indication of a slow network or client.
Thread Concurrency	Meaningful on Solaris 2.x only. Provides an indication of the level of thread concurrency.
Databases in use	Total number of databases being serviced by the server.

Connection Status

The Connection Status table in Directory Server console provides the following information about the amount of resources in use by each currently open connection:

Table 12-3 Server Performance Monitoring - Connection Status Table

Table Header	Description
Time opened	Indicates the time on the server when the connection was initially opened.
Started	Indicates the number of operations initiated by this connection.
Completed	Indicates the number of operations completed by the server for this connection.
Bound as	Indicates the distinguished name used by the client to bind to the server. If the client has not authenticated to the server, the server displays `not bound` in this field.
Read / Write	Indicates whether the server is currently blocked for read or write access to the client. Possible values include: • Not blocked. Indicates that the server is idle, actively sending data to the client, or actively reading data from the client. • Blocked. Indicates that the server is trying to send data to the client or read data from the client, but cannot. The probable cause is a slow network or client.

Global Database Cache Information

The Global Database Cache Information table in the Directory Server Console contains the following information:

Table 12-4 Server Performance Monitoring - Global Database Cache Table

Table Header	Description
Hits	Indicates the number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
Tries	The total number of requests performed on your directory since server startup.
Hit Ratio	The ratio of cache tries to successful cache hits. The closer this number is to 100% the better.
Pages read in	Indicates the number of pages read from disk into the cache.
Pages written out	Indicates the number of pages written from the cache back to disk.

Table 12-4 Server Performance Monitoring - Global Database Cache Table *(Continued)*

Table Header	Description
Read-only page evicts	Indicates the number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
Read-write page evicts	Indicates the number of read-write pages discarded from the cache to make room for new pages. This value differs from Pages Written Out in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.

Monitoring Your Server From the Command Line

You can monitor your directory server's current activities from any LDAP client by performing a search operation with the following characteristics:

- Search for attribute `objectClass=*`
- Search base: `cn=monitor`
- Search scope: `base`

For example:

```
ldapsearch -h directory.siroe.com -s base -b "cn=monitor"
"(objectclass=*)"
```

For information on searching the Directory Server, see "Using ldapsearch," on page 486.

The monitoring attributes for your server are found in the `cn=monitor,cn=config` entry.

When you monitor your server's activities using `ldapsearch`, you see the following information:

- `version:` Identifies the directory's current version number.
- `threads:` Current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.

- `connection:`*fd*`:`*opentime*`:`*opsinitiated*`:`*opscompleted*`:`*binddn*`:[rw]:` Provides the following summary information for each open connection (only available if you bind to the directory as the Directory Manager):
    - `fd`—The file descriptor used for this connection.
    - `opentime`—The time this connection was opened.
    - `opsinitiated`—The number of operations initiated by this connection.
    - `opscompleted`—The number of operations completed.
    - `binddn`—The distinguished name used by this connection to connect to the directory.
    - `rw`—The field shown if the connection is blocked for read or write.

  By default, this information is available to you only if you bind to the directory as the Directory Manager. However, you can change the ACI associated with this information to allow others to access the information.
- `currentconnections:` Identifies the number of connections currently in service by the directory.
- `totalconnections:` Identifies the number of connections handled by the directory since it started.
- `dtablesize:`Shows the number of file descriptors available to the directory. Each connection requires one file descriptor: one for every open index, one for log file management, and one for `ns-slapd` itself. Essentially, this value lets you know about how many more concurrent connections can be serviced by the directory. For more information on file descriptors, refer to your operating system documentation.
- `readwaiters:` Identifies the number of threads waiting to read data from a client.
- `opsinitiated:` Identifies the number of operations the server has initiated since it started.
- `opscompleted:` Identifies the number of operations the server has completed since it started.
- `entriessent:` Identifies the number of entries sent to clients since the server started.
- `bytessent:` Identifies the number of bytes sent to clients since the server started.

- `currentime:` Identifies the time when this snapshot of the server was taken. The time is displayed in Greenwich mean time (GMT) in UTC format.
- `starttime:` Identifies the time when the server started. The time is displayed in Greenwich mean time (GMT) in UTC format.
- `nbackends:` Identifies the number of back ends (databases) the server services.
- `concurrency:` Solaris 2.x only. Indicates the current level of thread concurrency.
- `backendmonitordn`: Identifies the DN of each directory database.

Monitoring Database Activity

You can monitor your database's current activities from Directory Server Console or from the command line. This section contains the following information:

- "Monitoring Database Activity From the Server Console," on page 391
- "Monitoring Databases From the Command Line," on page 395

Monitoring Database Activity From the Server Console

This section describes how you can use Directory Server Console to view the database performance monitors and what sort of information the performance monitors provide.

Viewing Database Performance Monitors

To monitor your database's activities:

1. On the Directory Server Console, select the Status tab. In the navigation tree, expand the Performance Counters folder and select the database that you want to monitor.

   The tab displays current information about database activity. If the server is currently not running, this tab will not provide performance monitoring information.

2. Click Refresh to refresh the currently displayed information. If you want the directory to continuously update the displayed information, select the Continuous checkbox and then click Refresh.

Overview of Database Performance Monitor Information

The directory provides database monitoring information as described in the following sections:

- General Information (Database)
- Summary Information Table
- Database Cache Information Table
- Database File-Specific Table

General Information (Database)

The directory provides the following general database information:

- Database

  Identifies the type of database that you are monitoring.

- Configuration DN

  Identifies the distinguished name that you must use as a search base to obtain these results using the `ldapsearch` command-line utility.

Summary Information Table

The Summary Information table provides the following information:

Table 12-5 Database Performance Monitoring - Summary Information

Performance Metric	Current Total
Readonly status	Indicates whether the database is currently in read-only mode. Your database is in read-only mode when the `readonly` attribute is set to `on`.
Entry cache hits	Indicates the total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
Entry cache tries	Indicates the total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against your server since server startup.

Table 12-5 Database Performance Monitoring - Summary Information *(Continued)*

Performance Metric	**Current Total**
Entry cache hit ratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100% the better. Whenever a search operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases and directory search performance drops. To improve this ratio, you can increase the number of entries that the directory maintains in the entry cache by increasing the value of the “Maximum Entries in Cache” attribute. See “Tuning Database Performance,” on page 410 for information on changing this value using the Server Console.
Current entry cache size (in bytes)	Indicates the total size of directory entries currently present in the entry cache.
Maximum entry cache size (in bytes)	Indicates the size of the entry cache maintained by the directory. This value is managed by the “Maximum Cache Size” attribute. See “Tuning Database Performance,” on page 410 for information on changing this value using the Server Console.
Current entry cache size (in entries)	Indicates the total number of directory entries currently present in the entry cache.
Maximum entry cache size (in entries)	Indicates the maximum number of directory entries that can be maintained in the entry cache. This value is managed by the “Maximum Entries in Cache” attribute. See “Tuning Database Performance,” on page 410 for information on changing this value using the Server Console.

Database Cache Information Table

The Database Cache Information table provides the following caching information:

Table 12-6 Database Performance Monitoring - Database Cache Information

Performance Metric	**Current Total**
Hits	Indicates the number of times the database cache successfully supplied a requested page. A page is a buffer of the size 2K.
Tries	Indicates the number of times the database cache was asked for a page.

Table 12-6 Database Performance Monitoring - Database Cache Information *(Continued)*

Performance Metric	**Current Total**
Hit ratio	Indicates the ratio of database cache hits to database cache tries. The closer this value is to 100%, the better. Whenever a directory operation attempts to find a portion of the database that is not present in the database cache, the directory has to perform a disk access to obtain the appropriate database page. Thus, as this ratio drops towards zero, the number of disk accesses increases and directory performance drops. To improve this ratio, you can increase the amount of data that the directory maintains in the database cache by increasing the value of the "Maximum Cache Size" attribute. See "Tuning Database Performance," on page 410 for information on changing this value using the Server Console.
Pages read in	Indicates the number of pages read from disk into the database cache.
Pages written out	Indicates the number of pages written from the cache back to disk. A database page is written to disk whenever a read-write page has been modified and then subsequently deleted from the cache. Pages are deleted from the database cache when the cache is full and a directory operation requires a database page that is not currently stored in cache.
Read-only page evicts	Indicates the number of read-only pages discarded from the cache to make room for new pages.
Read-write page evicts	Indicates the number of read-write pages discarded from the cache to make room for new pages. This value differs from Pages Written Out in that these are discarded read-write pages that have not been modified.

Database File-Specific Table

The directory displays a table for each index file that makes up your database. Each of the tables provides the following information:

Table 12-7 Database Performance Monitoring - Database File-Specific table

Performance Metric	**Current Total**
Cache hits	Number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file and the directory obtains the required data from the cache.
Cache misses	Number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed and the required data could not be found in the cache.
Pages read in	Indicates the number of pages brought to the cache from this file.

Table 12-7 Database Performance Monitoring - Database File-Specific table *(Continued)*

Performance Metric	**Current Total**
Pages written out	Indicates the number of pages for this file written from cache to disk.

Monitoring Databases From the Command Line

You can monitor your directory's database activities from any LDAP client by performing a search operation with the following characteristics:

- Search for attribute `objectClass=*`
- Search base: `cn=monitor,cn=`*database_instance*`,cn=ldbm database, cn=plugins, cn=config,` where database is the name of the database that you want to monitor
- Search scope: `base`

For example:

```
ldapsearch -h directory.siroe.com -s base -b
"cn=monitor,cn=Siroe,cn=ldbm database,cn=plugins, cn=config"
"objectclass=*"
```

In this example, the ldapsearch operation looks for the Siroe database. For information on searching the directory, see "Using ldapsearch," on page 486.

When you monitor your server's activities, you see the following information:

- `database:` Identifies the type of database you are currently monitoring.
- `readonly:` Indicates whether the database is in read-only mode. `0` indicates that the server is not in read-only mode, `1` indicates that it is in read-only mode.
- `entrycachehits:` Provides the same information as described in Entry cache hits in Table 12-5 on page 392.
- `entrycachetries:` Provides the same information as described in Entry cache tries in Table 12-5 on page 392.
- `entrycachehitratio:` Provides the same information as described in "Entry cache hit ratio," on page 393 in Table 12-5.
- `currententrycachesize:` Provides the same information as described in "Current entry cache size (in entries)," on page 393 in Table 12-5.

- `maxentrycachesize:` Provides the same information as described in "Maximum entry cache size (in entries)," on page 393 in Table 12-5.
- `dbchehits:` Provides the same information as described in Hits in Table 12-6 on page 393.
- `dbcachetries:` Provides the same information as described in Tries in Table 12-6 on page 393.
- `dbcachehitratio:` Provides the same information as described in Hit ratio in Table 12-6 on page 393.
- `dbcachepagein:` Provides the same information as described in Pages read in in Table 12-6 on page 393.
- `dbcachepageout:` Provides the same information as described in Pages written out in Table 12-6 on page 393.
- `dbcacheroevict:` Provides the same information as described in Read-only page evicts in Table 12-6 on page 393.
- `dbcacherwevict:` Provides the same information as described in Read-write page evicts in Table 12-6 on page 393.

Next the following information for each file that makes up your database is displayed:

- `dbfilename-`*number*: Indicates the name of the file. *number* provides a sequential integer identifier (starting at 0) for the file. All associated statistics for the file are given this same numerical identifier.
- `dbfilecachehit-`*number*: Provides the same information as described in Cache hits in Table 12-7 on page 394.
- `dbfilecachemiss-`*number*: Provides the same information as described in Cache misses in Table 12-7 on page 394.
- `dbfilepagein-`*number*: Provides the same information as described in Pages read in in Table 12-7 on page 394.
- `dbfilepageout-`*number*: Provides the same information as described in Pages written out in Table 12-7 on page 394.

Monitoring Database Link Activity

You can monitor the activity of your database links from the command line using the monitoring attributes. Use the `ldapsearch` command-line utility to return the attribute values that interest you. The monitoring attributes are stored in the following entry: `cn=monitor,cn=`*database_link_name*`,cn=chaining database,cn=plugins, cn=config`.

For example, you can use the `ldapsearch` command-line utility to retrieve the number of add operations received by a particular database link called DBLink1. First, type the following to change to the directory containing the utility:

```
cd /usr/iplanet/servers/shared/bin
```

Then, run `ldapsearch` as follows:

```
ldapsearch -h directory.siroe.com -p 389 -D "cn=Directory Manager"
-w secret -s sub -b "cn=monitor,cn=DBLink1,cn=chaining
database,cn=plugins,cn=config" "(objectclass=*)" nsAddCount
```

NOTE The above command should be typed on a single line. It does not appear on one line here because of page size constraints.

You can search for the following database link monitoring attributes:

Table 12-8 Database Link Monitoring Attributes

Attribute Name	Description
`nsAddCount`	Number of add operations received.
`nsDeleteCount`	Number of delete operations received.
`nsModifyCount`	Number of modify operations received.
`nsRenameCount`	Number of rename operations received.
`nsSearchBaseCount`	Number of base level searches received.
`nsSearchOneLevelCount`	Number of one-level searches received.
`nsSearchSubtreeCount`	Number of subtree searches received.
`nsAbandonCount`	Number of abandon operations received.
`nsBindCount`	Number of bind request received.
`nsUnbindCount`	Number of unbinds received.
`nsCompareCount`	Number of compare operations received.

Table 12-8 Database Link Monitoring Attributes

Attribute Name	**Description**
`nsOperationConnectionCount`	Number of open connections for normal operations.
`nsBindConnectionCount`	Number of open connections for bind operations.

For more information about `ldapsearch`, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

Chapter 13

Monitoring Directory Server Using SNMP

The server and database activity monitoring log setup described in Chapter 12, "Monitoring Server and Database Activity" is specific to iPlanet Directory Server. You can also monitor your Directory Server using the Simple Network Management Protocol (SNMP) which is a management protocol used for monitoring network activity which can be used to monitor a wide range of devices in real time.

SNMP has become interoperable on account of its widespread popularity. It is this interoperability combined with the fact that SNMP can take on numerous jobs specific to a whole range of different device classes, that make SNMP the ideal standard mechanism for global network control and monitoring. SNMP allows network administrators to unify all network monitoring activities, with Directory Server monitoring just part of the broader picture.

This chapter contains the following topics:

- About SNMP
- Overview of the Directory Server Management Information Base
- Setting Up SNMP
- Configuring SNMP for the Directory Server

About SNMP

SNMP is a protocol used to exchange data about network activity. With SNMP, data travels between a managed device and a network management station (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. iPlanet Directory Server has a subagent. The master agent exchanges information between the various subagents and the NMS. The master agent runs on the same host machine as the subagents it talks to.

You can have multiple subagents installed on a host machine. For example, if you have Directory Server, Enterprise Server, and Messaging Server all installed on the same host, the subagents for each of these servers communicates with the same master agent. In the Windows NT environment, the master agent is the SNMP service provided by the Windows NT operating system. In the UNIX environment, the master agent is installed with the iPlanet Administration Server.

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 Overview

SNMP exchanges network information in the form of *protocol data unit* (PDUs). PDUs contain information about variables stored on the managed device. These variables, also known as managed objects, have values and titles that are reported to the NMS as necessary. Communication between an NMS and a managed device takes place in one of two ways:

- NMS-Initiated Communication
- Managed Device-Initiated Communication

NMS-Initiated Communication

NMS-initiated communication is the most common type of communication between an NMS and a managed device. In this type of communication, the NMS either requests information from the managed device or changes the value of a variable stored on the managed device.

These are the steps that make up an NMS-initiated SNMP session:

1. The NMS determines which managed devices and objects need to be monitored.
2. The NMS sends a protocol data unit to the managed device's subagent through the master agent. This protocol data unit either requests information from the managed device or tells the subagent to change the values for variables stored on the managed device.
3. The subagent for the managed device receives the protocol data unit from the master agent.
4. If the protocol data unit from the NMS is a request for information about variables, the subagent gives information to the master agent and the master agent sends it back to the NMS in the form of another protocol data unit. The NMS then displays the information textually or graphically.

   If the protocol data unit from the NMS requests that the subagent set variable values, the subagent sets these values.

Managed Device-Initiated Communication

This type of communication occurs when the managed device needs to inform the NMS of an event that has occurred. A managed device initiates communication with an NMS to inform the NMS of a shut down or start up. Communication initiated by a managed device is also known as a *trap*. Directory Server sends a trap to the NMS whenever the Directory Server starts or stops.

These are the steps that make up a managed device-initiated SNMP session:

1. An event occurs on the managed device.
2. The subagent informs the master agent of the event.
3. The master agent sends a PDU to the NMS to inform the NMS of the event.
4. The NMS displays the information textually or graphically.

Overview of the Directory Server Management Information Base

Each iPlanet server has its own MIB. The Directory Server's MIB is a file called `netscape-ldap.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 network management software, such as HP OpenView, you can monitor your directory like all other managed devices on your network.

The directory MIB has the following object identifier: `iso.org.dod.internet.private.enterprises.netscape.nsldap` (`nsldapd OBJECT IDENTIFIER ::= { 1.3.6.1.4.1.1450.7 }`).

The object identifier is located in the `/usr/iplanet/servers/plugins/snmp` directory.

You can see administrative information about your directory and monitor the server in real-time using the directory MIB. The directory MIB is broken into three distinct tables of managed objects:

- Operations Table
- Entries Table
- Interaction Table

NOTE Before you can use the directory's MIB, you must compile it along with the MIBs that you will find in the following default location:

`/usr/iplanet/servers/plugins/snmp/mibs`

For information on how to compile MIBs, see your SNMP product documentation. The following sections describe the each table in detail.

About the Operations Table

The Operations Table provides statistical information about Directory Server access, operations, and errors. Table 13-1 on page 403 describes the managed objects stored in the Operations Table of the `netscape-ldap.mib` file.

Table 13-1 Operations Table Managed Objects and Descriptions

Managed Object	Description
dsAnonymousBinds	The number of anonymous binds to the directory since server startup.
dsUnauthBinds	The number of unauthenticated binds to the directory since server startup.
dsSimpleAuthBinds	The number of binds to the directory that were established using a simple authentication method (such as password protection) since server startup.
dsStrongAuthBinds	The number of binds to the directory that were established using a strong authentication method (such as SSL or an SASL mechanism like Kerberos) since server startup.
dsBindSecurityErrors	The number of bind requests that have been rejected by the directory due to authentication failures or invalid credentials since server startup.
dsInOps	The number of operations forwarded to this directory from another directory since server startup.
dsReadOps	The number of read operations serviced by this directory since application start. The value of this object will always be 0 because LDAP implements read operations indirectly via the search operation.
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.

Table 13-1 Operations Table Managed Objects and Descriptions *(Continued)*

Managed Object	Description
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.

The Entries Table

The Entries Table provides information about the contents of the directory entries. Table 13-2 describes the managed objects stored in the Entries Table in the `netscape-ldap.mib` file.

Table 13-2 Entries Table Managed Objects and Descriptions

Managed Object	Description
dsMasterEntries	The number of directory entries for which this directory contains the master entry. The value of this object will always be `0` (as no updates are currently performed).
dsCopyEntries	The number of directory entries for which this directory contains a slave copy. The value of this object will always be `0` (as no updates are currently performed).
dsCacheEntries	The number of entries cached in the directory.
dsCacheHits	The number of operations serviced from the locally held cache since application startup.
dsSlaveHits	The number of operations that were serviced from locally held replications (shadow entries). The value of this object will always be `0`.

Setting Up SNMP

The steps for configuring SNMP monitoring for your directory depend on whether you run your directory on Windows NT, UNIX or AIX. This section contains the following procedures :

- Setting Up SNMP on Windows NT
- Setting Up SNMP on UNIX
- Configuring the AIX SNMP Daemon

Setting Up SNMP on Windows NT

To set up SNMP support for Directory Server on a Windows NT machine:

1. Install the SNMP service on your NT server.

   Refer to your Windows NT operating system documentation for instructions.

2. Enable Directory Server statistics collection.

   See “Configuring SNMP for the Directory Server,” on page 408 for information.

3. Restart the Windows NT SNMP service.

   See “Starting and Stopping the SNMP Service on Windows NT,” on page 407 for information.

Setting Up SNMP on UNIX

To set up SNMP support for your Directory Server on a UNIX machine:

1. Configure and start the master agent using the Administration Server Console.

NOTE	If you are using the default port settings (161 for SNMP and 199 for SMUX) then you need to be root user. If you reconfigure the master agent configuration and have ports with values higher than 1000, then it is not necessary to be root user.

   For information on setting up the master agent, refer to *Managing Servers with iPlanet Console*.

2. On AIX machines, configure the AIX SNMP Daemon.

   See "Configuring the AIX SNMP Daemon," on page 406 for information.

3. Enable the directory subagent.

   See "Configuring SNMP for the Directory Server," on page 408 for information.

4. Start the directory subagent.

   See "Starting and Stopping the SNMP Subagent on UNIX," on page 407 for information.

Configuring the AIX SNMP Daemon

If your SNMP daemon is running on AIX, it supports SMUX . For this reason, you do not need to install a master agent. However, you need to change the AIX SNMP daemon configuration.

AIX uses several configuration files to filter its communications. One of them, `snmpd.conf`, needs to be changed so that the SNMP daemon accepts the incoming messages from the SMUX subagent. For more information, see the online manual page for `snmpd.conf`. You need to add a line to define each subagent.

For example, you might add this line to the `snmpd.conf`:

`smux 1.3.6.1.4.1.1.1450.7 ""` *IP_address net_mask*

where *IP_address* is the IP address of the host the subagent is running on, and *net_mask* is the network mask of the host.

NOTE Do not use the loopback address 127.0.0.1; use the real IP address instead.

If you need more information, see your related system documentation.

Starting and Stopping the SNMP Subagent on UNIX

To start, stop, and restart the SNMP subagent for a directory running on UNIX:

1. On the Directory Server Console, select the Configuration tab and then select the top most entry in the navigation tree in the left pane.
2. Select the SNMP tab in the right pane.
3. Click Start to start the subagent, click Stop to stop the subagent, or click Restart to restart the subagent.

   Stopping the directory does not stop the directory subagent. If you want to stop the subagent, you must do so from this tab.

NOTE If you add another server instance and you want the instance to be part of the SNMP network, you must restart the subagent.

Starting and Stopping the SNMP Service on Windows NT

It is important to note that the master agent on Windows NT is the SNMP Service and *not* the SNMP subagent as is the case on other platforms. The SNMP Service is installed and configured via the Windows NT control panel. For a directory running on Windows NT, the SNMP subagent is a DLL which the SNMP service invokes, and it is by using the information stored in the registry that the SNMP Service knows which subagent to load.

To start, stop and restart the SNMP subagent for a directory running on Windows NT :

1. Open the Control Panel and select Services.
2. Select SNMP from the Service list.
3. Click Start to start the SNMP Service, click Stop to stop the SNMP Service, or click Stop then Start to restart the SNMP Service.

   Stopping the directory does not stop the directory subagent. If you want to stop the subagent, you must do so from the Control Panel.

NOTE If you add another server instance and you want the instance to be part of the SNMP network, you must restart the subagent.

Configuring SNMP for the Directory Server

To configure SNMP settings from the Directory Server Console:

1. Make sure the Directory Server is running.
2. On 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 SNMP tab in the right pane.
4. Select the “Enable Statistics Collection” checkbox to enable Directory Server statistics collection. Clear the checkbox to disable it.
5. For UNIX servers, enter the hostname on which the master agent resides and the port number used to communicate with the master agent in the Master Host and Master Port text boxes.

   NOTE The hostname and port number are required.

   The defaults are `localhost` and `199` respectively.
6. Enter a description that uniquely describes the directory instance in the Description text box.
7. Type the name the company or organization to which the directory belongs in the Organization text box.
8. Type the location within the company or organization where the directory resides in the Location text box.
9. Type the email address of the person responsible for maintaining the directory in the Contact text box.
10. Click Save.
11. Restart the subagent (UNIX), or restart the SNMP service (Windows NT).

    See “Starting and Stopping the SNMP Subagent on UNIX,” on page 407 or “Starting and Stopping the SNMP Service on Windows NT,” on page 407 for information as appropriate.

Chapter 14

Tuning Directory Server Performance

This chapter describes the tools provided with iPlanet Directory Server to help optimize performance. It also provides tips to improve the performance of your directory

This chapter contains the following sections:

- Tuning Server Performance
- Tuning Database Performance

Tuning Server Performance

You can manage your server's performance by limiting the amount of resources the server uses to proces client search requests. You can define:

- The maximum number of entries the server returns to the client in response to a search operation (size limit attribute)
- The maximum amount of real time (in seconds) you want the server to spend performing a search request (time limit attribute)
- The time (in seconds) during which the server maintains an idle connection before terminating it (idle timeout attribute)
- The maximum number of file descriptors available to the directory server (max number of file descriptors attribute)

To configure Directory Server to optimize performance:

1. On the Directory Server Console, select the Configuration tab and then select the topmost entry in the navigation tree in the left pane.

   The tabs that are displayed in the right pane control server-wide configuration attributes.

2. Select the Performance tab in the right pane.

   The current server performance settings appear.

3. Set the maximum number of entries the server will return to the client in response to a search operation by entering a new value in the Size Limit text box.

   If you do not want to set a limit, type `-1` in this text box.

4. Enter the maximum amount of real time (in seconds) you want the server to spend performing a search request in the Time Limit text box.

   If you do not want to set a limit, type zero (`0`) in this text box.

5. Enter the time (in seconds) during which you want the server to maintain an idle connection before terminating it in the Idle Timeout text box.

   If you do not want to set a limit, type zero (`0`) in this text box.

6. Set the maximum number of file descriptors available to the directory server in the Max Number of File Descriptors text box.

   This option is not available on the Windows NT and IBM AIX platforms. For more information on this parameter, see the *iPlanet Directory Server Configuration, Command, and File Reference*.

For a better understanding of how these parameters impact your server's search performance, see "About Indexes," on page 327.

Tuning Database Performance

The following sections describe methods for tuning database performance:

- Optimizing Search Performance
- Tuning Transaction Logging

Optimizing Search Performance

You can improve server performance on searches by tuning database settings. The database attributes that affect performance mainly define the amount of memory available to the server.

To improve the cache hit ratio on search operations, you can increase the amount of data that the directory server maintains in the database cache. Do this by increasing the number of entries stored in the cache, and by increasing the cache size. The maximum values that you can set for these attributes depends on the amount of real memory on your machine. Roughly, the amount of available memory on your machine should always be greater than:

```
(Maximum Entries in Cache + Maximum Cache Size) x average entry size
```

Use caution when changing these two attributes. Your ability to improve server performance with these attributes depends on the size of your database, the amount of physical memory available on your machine, and whether directory searches are random (that is, if your directory clients are searching for random and widely scattered directory data).

If your database does not fit into memory and if searches are random , attempting to increase the values set on these attributes does not help directory performance. In fact, changing these attributes may harm overall performance.

You can tune the following attributes:

- The attributes of the database that manages all other database instances. In Directory Server Console, you can see only the databases that contain your directory data, and the `NetscapeRoot` database. However, the server uses another database to manage these. On this database, you can change the following attributes to improve performance:
    - The amount of memory that you want to make available for all databases (maximum cache size attribute)
    - The maximum number of entries you want the server to verify in response to a search request (look-through limit attribute)
- The attributes of each database that you use to store directory data, including the server configuration data in the `NetscapeRoot` database. On these databases, you can change the following attributes to improve performance:
    - The maximum number of entries you want the server to keep in memory (maximum entries in cache attribute)
    - The amount of memory you want to make available for cached entries (memory available for cache attribute)

To configure the default database attributes that apply to all other database instances:

1. On the Directory Server Console, select the Configuration tab, then in the navigation tree expand the Data Icon, and highlight the Database Settings node.

   This displays the Database tabs in the right pane.

2. Select the LDBM Plugin Settings tab in the right pane.

   This tab contains the database attributes for all databases stored on this server.

3. In the Maximum Cache Size field, enter a value corresponding to the amount of memory that you want to make available for all databases.

4. In the look-through limit field, enter the maximum number of entries you want the server to check in response to a search request.

   If you do not want to set a limit, type **`-1`** in this text box. If you bind to the directory as the Directory Manager, by default the look-through limit is unlimited, and overrides any settings you specify here.

To configure the attributes of each database that stores your directory data:

1. On the Directory Server Console, select the Configuration tab, then in the navigation tree expand the Data Icon. Expand the suffix of the database you want to tune and highlight the database.

   The tabs displayed in the right pane control parameter settings for this database.

2. Select the Database Settings tab in the right pane.

3. Enter the number of entries you want the server to keep in memory in the Maximum Entries in Cache field.

4. Enter the amount of memory you want to make available for cached entries in the Memory Available for Cache field.

   If you are creating a very large database from LDIF, set this attribute as large as possible, depending on the memory available on your machine. The larger this parameter, the faster your database will be created.

   When you have finished creating your database, be sure to set this parameter back to some lower value before you run your server in a production environment.

Tuning Transaction Logging

Every Directory Server contains a transaction log which logs operations for all the databases it manages. Whenever a directory database operation such as a write is performed, the server logs the operation to the transaction log. For best performance, the directory does not perform the operation immediately. Instead, the operation is stored in a temporary memory cache on the directory server until the operation is completed.

If the server experiences a failure, such as a power outage, and shuts down abnormally, the information about recent directory changes that were stored in the cache are lost. However, when the server restarts, the directory automatically detects the error condition and uses the database transaction log to recover the database.

Although database transaction logging and database recovery are automatic processes that require no intervention, you may want to tune some of the database transaction logging attributes to optimize performance. The following sections describe the options:

- "Changing the Location of the Database Transaction Log," on page 414
- "Changing the Database Checkpoint Interval," on page 414
- "Disabling Durable Transactions," on page 415

CAUTION The transaction logging attributes are provided only for system modifications and diagnostics. These settings should be changed only with the guidance of iPlanet Professional Services or iPlanet Support.

Setting these attributes and other configuration attributes inconsistently may cause the directory to be unstable.

Changing the Location of the Database Transaction Log

By default, the database transaction log file is stored in the `/usr/iplanet/servers/slapd-`*`serverID`*`/db` directory along with the database files themselves. Because the purpose of the transaction log is to aid in the recovery of a directory database that was shut down abnormally, it is a good idea to store the database transaction log on a different disk from the one containing the directory database. Storing the database transaction log on a separate physical disk may also improve directory performance.

To change the location of the database transaction log file, use the following procedure:

1. Stop the Directory Server.

   For instructions, refer to "Starting/Stopping the Server From the Console," on page 29.

2. Use the `ldapmodify` command-line utility to add the `nsslapd-db-logdirectory` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry. Provide the full path to the log directory in the attribute.

   For information on the `nsslapd-db-logdirectory` attribute syntax, see the *iPlanet Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 49.

3. Restart Directory Server.

Changing the Database Checkpoint Interval

At regular intervals, the directory server writes operations logged in the transaction log to the disk, and logs a checkpoint entry in the database transaction log. By indicating which changes have already been written to the directory, checkpoint entries indicate where to begin recovery from the transaction log, thus speeding up the recovery process.

By default, the directory server is set up to send a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval may increase the performance of directory write operations. However, increasing the checkpoint interval may also increase the amount of time required to recover directory databases after a disorderly shutdown and require more disk space due to large database transaction log files. Therefore, you should only modify only this attribute if you are familiar with database optimization and can fully assess the effect of the change.

To modify the checkpoint interval, use the following procedure:

1. Stop the Directory Server.

   For instructions, refer to "Starting/Stopping the Server From the Command Line," on page 30.

2. Use the `ldapmodify` command-line utility to add the `nsslapd-db-checkpoint-interval` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry.

   For more information on the syntax of the `nsslapd-db-checkpoint-interval` attribute, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 49.

3. Restart the Directory Server.

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 enabled, every directory database operation is written to the database transaction log file, but may not be physically written to disk immediately. If a directory change was written to the logical database transaction log file but not physically written to disk at the time of a system crash, you cannot recover the change. When durable transactions are disabled, the recovered database is consistent, but does not reflect the results of any LDAP write operations that completed just before the system crash.

By default, durable database transaction logging is enabled. To disable durable transaction logging, use the following procedure:

1. Stop the Directory Server.

   For instructions, refer to "Starting/Stopping the Server From the Command Line," on page 30.

2. Use the `ldapmodify` command-line utility to add the `nsslapd-db-durable-transactions` attribute to the `cn=config,cn=ldbm database,cn=plugins,cn=config` entry, and set the value of this attribute to **`off`**.

   For information on the syntax of the `nsslapd-db-durable-transactions` attribute, see the *iPlanet Directory Server Configuration, Command, and File Reference*. For instructions on using `ldapmodify`, refer to "Adding and Modifying Entries Using ldapmodify," on page 49.

3. Restart the Directory Server.

iPlanet Plug-Ins Reference

Chapter 15, “Administering Directory Server Plug-Ins”

Chapter 16, “Using the Pass-Through Authentication Plug-In”

Chapter 17, “Using the Attribute Uniqueness Plug-In”

Chapter 15

Administering Directory Server Plug-Ins

Directory Server plug-ins extend the functionality of the server. iPlanet Directory Server ships with several plug-ins to help you manage your directory. This chapter contains general information on the types of plug-ins available, and how to enable or disable them. This chapter is divided into the following sections:

- Server Plug-in Functionality Reference
- Enabling and Disabling Plug-Ins From the Server Console

Server Plug-in Functionality Reference

The following tables provide you with a quick overview of the plug-ins provided with iPlanet Directory Server 5.0, along with their configurable options, configurable arguments, default setting, dependencies, general performance related information and further reading. These tables will allow you to weigh up plug-in performance gains and costs and choose the optimal settings for your deployment. The Further Information heading cross references further reading where this is available.

7-bit Check Plug-In

Plug-in Name	7-bit check (NS7bitAtt)
DN of Configuration Entry	`cn=7-bit check,cn=plugins,cn=config`
Description	Checks certain attributes are 7-bit clean
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	list of attributes (`uid mail userpassword`) followed by "," and then suffix(ex) on which the check is to occur
Dependencies	None
Performance Related Information	None
Further Information	If your Directory Server uses non-ASCII characters, for example, Japanese, turn this plug-in off.

ACL Plug-In

Plug-in Name	ACL Plugin
DN of Configuration Entry	`cn=ACL Plugin,cn=plugins,cn=config`
Description	ACL access check plug-in
Configurable Options	on | off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	N/A
Further Information	Chapter 6, “Managing Access Control.”

ACL Preoperation Plug-In

Plug-in Name	ACL preoperation
DN of Configuration Entry	`cn=ACL preoperation,cn=plugins,cn=config`
Description	ACL access check plug-in
Configurable Options	on | off
Default Setting	on
Configurable Arguments	None
Dependencies	database
Performance Related Information	None
Further Information	Chapter 6, “Managing Access Control.”

Binary Syntax Plug-In

Plug-in Name	Binary Syntax
DN of Configuration Entry	`cn=Binary Syntax,cn=plugins,cn=config`
Description	Syntax for handling binary data
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Boolean Syntax Plug-In

Plug-in Name	Boolean Syntax
DN of Configuration Entry	`cn=Boolean Syntax,cn=plugins,cn=config`
Description	Syntax for handling booleans
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Case Exact String Syntax Plug-In

Plug-in Name	Case Exact String Syntax
DN of Configuration Entry	`cn=Case Exact String Syntax,cn=plugins,cn=config`
Description	Syntax for handling case-sensitive strings
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Case Ignore String Syntax Plug-In

Plug-in Name	Case Ignore String Syntax
DN of Configuration Entry	`cn=Case Ignore String Syntax,cn=plugins,cn=config`
Description	Syntax for handling case-insensitive strings
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Chaining Database Plug-In

Plug-in Name	Chaining Databse
DN of Configuration Entry	`cn=Chaining database,cn=plugins,cn=config`
Description	Syntax for handling DNs
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 3, “Configuring Directory Databases.”

Class of Service Plug-In

Plug-in Name	Class of Service
DN of Configuration Entry	`cn=Class of Service,cn=plugins,cn=config`
Description	Allows for sharing of attributes between entries
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 5, “Advanced Entry Management.”

Country String Syntax Plug-In

Plug-in Name	Country String Syntax Plug-in
DN of Configuration Entry	`cn=Country String Syntax,cn=plugins,cn=config`
Description	Syntax for handling countries
Configurable Options	on | off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Distinguished Name Syntax Plug-In

Plug-in Name	Distinguished Name Syntax
DN of Configuration Entry	`cn=Distinguished Name Syntax,cn=plugins,cn=config`
Description	Syntax for handling DNs
Configurable Options	on | off
Default Setting	on
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Generalized Time Syntax Plug-In

Plug-in Name	Generalized Time Syntax
DN of Configuration Entry	`cn=Generalized Time Syntax,cn=plugins,cn=config`
Description	Syntax for dealing with dates, times and time zones
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	The Generalized Time String consists of the following: four digit year, two digit month (for example, 01 for January), two digit day, two digit hour, two digit minute, two digit second, an optional decimal part of a second and a time zone indication. We strongly recommend that you use the Z time zone indication which stands for Greenwich Mean Time.

Integer Syntax Plug-In

Plug-in Name	Integer Syntax
DN of Configuration Entry	`cn=Integer Syntax,cn=plugins,cn=config`
Description	Syntax for handling integers
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Internationalization Plug-In

Plug-in Name	Internationalization Plugin
DN of Configuration Entry	`cn=Internationalization Plugin,cn=plugins,cn=config`
Description	Syntax for handling DNs
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	The Internationalization has one argument which must not be modified: `/usr/iplanet/servers/slapd-`*serverID*`/config/slapd-collations.conf` This directory stores the collation orders and locales used by the internationalization plug-in.
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	See Appendix D, “Internationalization.”

ldbm Database Plug-In

Plug-in Name	ldbm database Plug-in
DN of Configuration Entry	`cn=ldbm database plug-in,cn=plugins,cn=config`
Description	Implements local databases
Configurable Options	N/A
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	See *iPlanet Directory Server Configuration, Command, and File Reference* for further information on database plug-in attributes.
Further Information	Chapter 3, “Configuring Directory Databases.”

Legacy Replication Plug-In

Plug-in Name	Legacy Replication plug-in
DN of Configuration Entry	`cn=Legacy Replication plug-in,cn=plugins,cn=config`
Description	Enables iPlanet Directory Server 5.0 to be a consumer of a 4.1 supplier
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None. This plug-in can be disabled if the server is not (and never will be) a consumer of a 4.1 server.
Dependencies	database
Performance Related Information	None
Further Information	Chapter 8, “Managing Replication.”

Multimaster Replication Plug-In

Plug-in Name	Multimaster Replication Plugin
DN of Configuration Entry	`cn=Multimaster Replication plugin,cn=plugins, cn=config`
Description	Enables replication between two 5.0 Directory Servers
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	databse
Performance Related Information	N/A
Further Information	You can turn this plug-in off if you only have one server which will never replicate. See also Chapter 8, “Managing Replication.”

Octet String Syntax Plug-in

Plug-in Name	Octet String Syntax
DN of Configuration Entry	`cn=Octet String Syntax,cn=plugins,cn=config`
Description	Syntax for handling octet strings
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

CLEAR Password Storage Plug-In

Plug-in Name	CLEAR
DN of Configuration Entry	`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. You should leave this plug-in running at all times.
Further Information	Chapter 7, “User Account Management.”

CRYPT Password Storage Plug-In

Plug-in Name	CRYPT
DN of Configuration Entry	`cn=CRYPT,cn=Password Storage Schemes,cn=plugins, cn=config`
Description	CRYPT password storage scheme used for password encryption
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 7, “User Account Management.”

NS-MTA-MD5 Password Storage Plug-In

Plug-in Name	NS-MTA-MD5
DN of Configuration Entry	`cn=NS-MTA-MD5,cn=Password Storage Schemes,cn=plugins, cn=config`
Description	NS-MTA-MD5 password storage scheme for password encryption
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. iPlanet recommends that you leave this plug-in running at all times.
Further Information	You cannot choose to encrypt passwords using the NS-MTA-MD5 password storage scheme. The storage scheme is present in iPlanet Directory Server 5.0 but only for reasons of backward compatibility with earlier versions of Directory Server. See Chapter 7, “User Account Management.”

SHA Password Storage Plug-In

Plug-in Name	SHA
DN of Configuration Entry	`cn=SHA,cn=Password Storage Schemes,cn=plugins,cn=config`
Description	SHA password storage scheme for password encryption
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	If your directory does not contain passwords encrypted using the SHA password storage scheme, you may turn this plug-in off. You should choose SSHA preferablly than SHA because SSHA is a far more secure option.
Further Information	Chapter 7, "User Account Management."

SSHA Password Storage Plug-in

Plug-in Name	SSHA
DN of Configuration Entry	`cn=SSHA,cn=Password Storage Schemes,cn=plugins,cn=config`
Description	SSHA password storage scheme for password encryption
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 7, "User Account Management."

Postal Address String Syntax Plug-In

Plug-in Name	Postal Address Syntax
DN of Configuration Entry	`cn=Postal Address Syntax,cn=plugins,cn=config`
Description	Syntax used for handling postal addresses
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

PTA Plug-In

Plug-in Name	Pass-Through Authentication Plugin
DN of Configuration Entry	`cn=Pass Through Authentication,cn=plugins,cn=config`
Description	Enables pass-through authentication, the mechanism which allows one directory to consult another to authenticate bind requests. This plug-in is not listed in Directory Server Console if you use the same server for your user directory and configuration directory.
Configurable Options	on | off
Default Setting	`off`
Configurable Arguments	`ldap://iplanet.com:389/o=iplanet`
Dependencies	None
Performance Related Information	Chapter 16, “Using the Pass-Through Authentication Plug-In.”
Further Information	Chapter 16, “Using the Pass-Through Authentication Plug-In.”

Referential Integrity Postoperation Plug-In

Plug-in Name	Referential Integrity Postoperation
DN of Configuration Entry	`cn=Referential Integrity Postoperation,cn=plugins, cn=config`
Description	Enables the server to ensure referential integrity
Configurable Options	All configuration and on | off
Default Setting	off
Configurable Arguments	When enabled the post operation Referential Integrity plug-in performs integrity updates on the `member`, `uniquemember`, `owner` and `seeAlso` attributes immediately after a delete or rename operation. You can reconfigure the plug-in to perform integrity checks on all other attributes. Configurable arguments are as follows: (1) Check for referential integrity `-1` = no check for referential integrity `0` = check for referential integrity is performed immediately `positive integer` = request for referential integrity is queued and processed at a later stage. This positive integer serves as a wake-up call for the thread to process the request, at intervals corresponding to the integer specified. (2) Log file for storing the change, for example `/usr/iplanet/logs/referint` (3) All the additional attrribute names you want to be checked for referential integrity.
Dependencies	database
Performance Related Information	You should enable the Referential Integrity plug-in on only one master in a multimaster replication environment to avoid conflict resolution loops. When enabling the plug-in on chained servers you must be sure to analyze your performance resource and time needs as well as your integrity needs.
Further Information	See “Maintaining Referential Integrity,” on page 64.

Retro Change Log Plug-In

Plug-in Name	Retro Changelog Plugin
DN of Configuration Entry	`cn=Retro Changelog Plugin,cn=plugins,cn=config`
Description	Used by LDAP clients for maintaining application compatibility with Directory Server 4.x versions. Maintains a log of all changes occuring in the Directory Server. The Retro Changelog offers the same functionality as the changelog in the 4.x versions of Directory Server.
Configurable Options	on | off
Default Setting	`off`
Configurable Arguments	See *iPlanet Directory Server Configuration, Command, and File Reference* for further information on the two configuration attributes for the retro change log plug-in.
Dependencies	None
Performance Related Information	May slow down Directory Server performance.
Further Information	Chapter 8, "Managing Replication."

Roles Plug-In

Plug-in Name	Roles Plugin
DN of Configuration Entry	`cn=Roles Plugin,cn=plugins,cn=config`
Description	Enables the use of roles in the Directory Server
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	Chapter 5, "Advanced Entry Management."

Telephone Syntax Plug-In

Plug-in Name	Telephone Syntax
DN of Configuration Entry	`cn=Telephone Syntax,cn=plugins,cn=config`
Description	Syntax for handling telephone numbers
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

UID Uniqueness Plug-in

Plug-in Name	UID Uniqueness plug-in
DN of Configuration Entry	`cn=UID Uniqueness,cn=plugins,cn=config`
Description	Checks that the values of specified attributes are unique each time a modification occurs on an entry.
Configurable Options	on | off
Default Setting	`off`

Configurable Arguments	Enter the following arguments:
	`uid`
	`"DN"`
	`"DN"...`
	if you want to check for uid attribute uniqueness in all listed subtrees.
	However, enter the following arguments:
	`attribute="uid"`
	`MarkerObjectclass = "ObjectClassName"`
	and optionally
	`requiredObjectClass = "ObjectClassName"`
	if you want to check for uid attribute uniqueness when adding or updating entries with the requiredObjectClass, starting from the parent entry containing the ObjectClass as defined by the MarkerObjectClass attribute.

Dependencies	N/A

Plug-in Name	UID Uniqueness plug-in
Performance Related Information	This plug-in may slow down Directory Server performance. In a multimaster replication environment, the UID Uniqueness plug-in will not work at all and should therefore not be enabled. If you try to add a new entry to a server where the UID Uniqueness plug-in is enabled and a referral has been created in a subtree, then the UID Uniqueness plug-in will not work. It will not work because if it sees any other error apart from noSuchObject (meaning that the entry does not already exist), which it will do if a referral is created, then it will return an operations error preventing you from adding your new entry. To prevent being blocked by such an operations error, disable the plug-in on the server where you created the referral. If, however, you still want to run a UID Uniqueness check, make sure that you only activate the plug-in on the last of the referred to servers to prevent it from blocking the referral mechanism.
Further Information	Chapter 17, “Using the Attribute Uniqueness Plug-In.”

URI Plug-in

Plug-in Name	URI Syntax
DN of Configuration Entry	`cn=URI Syntax,cn=plugins,cn=config`
Description	Syntax for handling URIs (Unique Resource Identifiers) including URLs (Unique Resource Locators)
Configurable Options	on | off
Default Setting	`on`
Configurable Arguments	None
Dependencies	None
Performance Related Information	Do not modify the configuration of this plug-in. You should leave this plug-in running at all times.
Further Information	

Enabling and Disabling Plug-Ins From the Server Console

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

1. On the Directory Server Console, select the Configuration tab.
2. Double-click the Plugins folder in the navigation tree.
3. Select the plug-in from the Plugins list.
4. To disable the plug-in, clear the “Enabled” checkbox. To enable the plug-in, check this checkbox.
5. Click Save.
6. Restart the directory server.

Chapter 16

Using the Pass-Through Authentication Plug-In

Pass-through authentication (PTA) is a mechanism by which one directory server consults another to authenticate bind requests. The PTA plug-in provides this functionality; allowing a directory server to accept simple bind operations (password based) for entries not stored in its local database.

iPlanet Directory Server 5.0 uses PTA to allow you to administer your user and configuration directories on separate instances of Directory Server.

This chapter describes the PTA plug-in in the following sections:

- How Directory Server 5.0 Uses PTA
- PTA Plug-In Syntax
- Configuring the PTA Plug-In
- PTA Plug-In Syntax Examples

How Directory Server 5.0 Uses PTA

If you install the configuration directory and the user directory on separate instances of Directory Server, the installation program automatically sets up PTA to allow the Configuration Administrator user (usually `admin`) to perform administrative duties.

PTA is required in this case because the `admin` user entry is stored under `o=NetscapeRoot` in the configuration directory. Therefore, attempts to bind to the user directory as `admin` would normally fail. PTA allows the user directory to transmit the credentials to the configuration directory which verifies them. The user directory then allows the `admin` user to bind.

The user directory in this example acts as the PTA directory, that is, the server that passes through bind requests to another directory server. The configuration directory acts as the *authenticating directory*, that is, the server that contains the entry and verifies the bind credentials of the requesting client.

You will also see the term *pass-through subtree* used in this chapter. The pass-through subtree is the subtree *not* present on the PTA directory. When a user's bind DN contains this subtree, the user's credentials are passed on to the authenticating directory.

NOTE The PTA plug-in is not listed in Directory Server Console when you use the same server for your user directory and your configuration directory.

Here's how pass-through authenticationworks:

1. You install the configuration directory server (authenticating directory) on Machine A.
   - Server Name: **`configdir.siroe.com`**
   - Suffix: **`o=NetscapeRoot`**
2. You install the user directory server (PTA directory) on Machine B.
   - Server Name: **`userdir.siroe.com`**
   - Suffix: **`dc=siroe,dc=com`**
3. During the installation of the user directory on Machine B, you are prompted to provide an LDAP URL. This URL points to the configuration directory on Machine A.
4. The installation program adds an entry to the `dse.ldif` file on the user directory that enables the PTA plug-in.

   This entry contains the LDAP URL you provided. For example:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
```

```
nsslapd-pluginarg0: ldap://config.siroe.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

The user directory is now configured to send all bind requests for entries whose DN contains `o=NetscapeRoot` to the configuration directory `configdir.siroe.com`.

5. When installation is complete, the `admin` user attempts to connect to the user directory to begin adding users.

6. The setup program adds the `admin` user's entry to the directory as `uid=admin, ou=TopologyManagement,o=NetscapeRoot`. So, the user directory passes the bind request through to the configuration directory as defined by the PTA plug-in configuration.

7. The configuration directory authenticates the user's credentials and sends the information back to the user directory.

8. The user directory allows the admin user to bind.

PTA Plug-In Syntax

PTA plug-in configuration information is specified in the `cn=Pass Through Authentication,cn=plugins,cn=config` entry in the `dse.ldif` file on the PTA directory (the user directory configured to pass through bind requests to the authenticating directory) using the syntax described in this section.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.extension
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: ldap|ldaps://authDS/subtree [maxconns,maxops,timeout,ldver,connlifetime]
```

The variable components of the PTA plug-in syntax are described in Table 16-1.

Notes:

- The LDAP URL (ldap|ldaps://*authDS*/*subtree*) must be separated from the optional parameters (*maxconns*, *maxops*, *timeout*, *ldver*, *connlifetime*) by a single space.
- If you explicitly define any of the optional parameters, you must define all of them, even if you specify only the default values.
- You can specify several authenticating directories or subtrees by incrementing the `nsslapd-pluginarg` attribute suffix by 1 each time. This is illustrated in "Specifying Multiple Authenticating Directory Servers," on page 451.

The optional parameters are described in the following table in the order inwhich they appear in the syntax.

Table 16-1 PTA Plug-In Parameters

Variable	**Definition**
state	Defines whether the plug-in is enabled or disabled. Acceptable values are **`on`** or **`off`** . See "Turning the Plug-in On or Off," on page 446 for more information.
extension	File extension for the plug-in. The extension is always `.sl` on HP-UX, `.so` on all other UNIX platforms, and `.dll` on Windows NT.
ldap|ldaps	Defines whether SSL is used for communication between the two directory servers. See "Configuring the Servers to Use a Secure Connection," on page 447 for more information.
authDS	The authenticating directory hostname. You can specify the port number of the directory server by adding a colon and then the port number. For example: `ldap://dirserver.siroe.com:390/` If you do not specify the port number, the PTA server attempts to connect using: • Port 389 if `ldap://` is specified in the URL. • Port 636 if `ldaps://` is specified in the URL. See "Specifying the Authenticating Directory Server," on page 448 for more information.
subtree	The pass-through subtree. The PTA directory server passes through bind requests to the authenticating directory server from all clients whose DN is in this subtree. See "Specifying the Pass-Through Subtree," on page 449 for more information.

Table 16-1 PTA Plug-In Parameters *(Continued)*

Variable	Definition
maxconns	Optional. The maximum number of connections the PTA directory can simultaneously open to the authenticating directory. The default is 3. See "Configuring the Optional Parameters," on page 449 for more information.
maxops	Optional. The maximum number of simultaneous operations (usually bind requests) the PTA directory can send to the authenticating directory within a single connection. The default is 5. See "Configuring the Optional Parameters," on page 449 for more information.
timeout	Optional. The time limit, in seconds, that the PTA directory waits for a response from the authenticating directory server. If this timeout is exceeded, the server returns an error to the client. The default is 300 seconds (five minutes). Specify zero (0) to indicate no time limit should be enforced. See "Configuring the Optional Parameters," on page 449 for more information.
ldver	Optional. The version of the LDAP protocol used to connect to the authenticating directory. iPlanet Directory Server supports LDAP version 2 and 3. The default is version 3. See "Configuring the Optional Parameters," on page 449 for more information.
connlifetime	Optional. The time limit, in seconds, within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating directory. The server will not close the connection unless a bind request is initiated and the directory determines the connection lifetime has been exceeded. If you do not specify this option, or if only one host is listed, no connection lifetime will be enforced. If two or more hosts are listed, the default is 300 seconds (five minutes). See "Configuring the Optional Parameters," on page 449 for more information.

Configuring the PTA Plug-In

The only method for configuring the PTA plug-in is to modify the entry `cn=Pass Through Authentication,cn=plugins,cn=config` in the `dse.ldif` file. To modify the `dse.ldif` file, you must proceed as follows:

1. Use the `ldapmodify` command to modify `cn=Pass Through Authentication,cn=plugins,cn=config`

2. Restart Directory Server.

Before you configure any of the parameters discussed in this section, the PTA plug-in entry must be present in the `dse.ldif` file. If this entry does not exist, you must create it with the appropriate syntax, as described in "PTA Plug-In Syntax," on page 443.

NOTE If you installed the user and configuration directories on different instances of the directory, the PTA plug-in entry is automatically added to the user directory's `dse.ldif` file. If you installed the user and configuration directories on the same instance, the syntax is not automatically added and you need to add it manually.

This section provides information about configuring the plug-in in the following sections:

- Turning the Plug-in On or Off
- Configuring the Servers to Use a Secure Connection
- Specifying the Authenticating Directory Server
- Specifying the Pass-Through Subtree
- Configuring the Optional Parameters

Turning the Plug-in On or Off

To turn the PTA plug-in on from the command line:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

2. Use the `ldapmodify` command to import the LDIF file into the directory.

For detailed information on the `ldapmodify` command, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

3. When you enable the plug-in, you must also check that the plug-in initialization function is properly defined.

   The entry `cn=Pass Through Authentication,cn=plugins,cn=config` should contain the following attribute-value pairs:

```
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.extension
nsslapd-pluginInitfunc: passthruauth_init
```

   where *extension* is always `.sl` on HP-UX, `.so` on all other UNIX platforms, and `.dll` on Windows NT.

4. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 29.

To disable the plug-in, change the LDIF update statements to delete the `nsslapd-pluginenabled: on` statement, and add the `nsslapd-pluginenabled: off` statement. Whenever you enable or disable the PTA plug-in from the command line, you must restart the server.

Configuring the Servers to Use a Secure Connection

You can configure the PTA directory to communicate with the authenticating directory over SSL. You do this by specifying LDAPS in the LDAP URL of the PTA directory.

To configure the PTA directory and authenticating directory to use SSL:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldaps://authDS/subtree [optional_parameters]
```

   For information on the variable components in this sytax, refer to "PTA Plug-In Parameters," on page 444.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 29.

Specifying the Authenticating Directory Server

The authenticating directory contains the bind credentials for the entry with which the client is attempting to bind. The PTA directory passes the bind request to the host you define as the authenticating directory. You specify the authenticating directory server by replacing *authDS* in the LDAP URL of the PTA directory with the authenticating directory's hostname.

To specify the authenticating directory for PTA:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: add
add: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://authDS/subtree [optional_parameters]
```

   Optionally, you can include a colon followed by a port number. If you do not specify the port number, the PTA directory server attempts to connect using:

   - Port 389 if ldap:// is specified in the URL.
   - Port 636 if ldaps:// is specified in the URL.

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

   `"ldap://dirserver.siroe.com:389/`*subtree* `[`*Parameters*`]"`

   For information on the variable components in this sytax, refer to "PTA Plug-In Parameters," on page 444.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 29.

Specifying the Pass-Through Subtree

The PTA directory passes through bind requests to the authenticating directory from all clients whose DN is defined in the pass-through subtree. You specify the subtree by replacing the *subtree* parameter in the LDAP URL of the PTA directory.

The pass-through subtree must not exist in the PTA directory. If it does, the PTA directory attempts to resolve bind requests using its own directory contents and the binds fail.

To specify the pass-through subtree:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: add
add: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://authDS/subtree [optional_parameters]
```

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

   `"ldap://dirserver.siroe.com/o=NetscapeRoot [`*Parameters*`]"`

   For information on the variable components in this sytax, refer to “PTA Plug-In Parameters,” on page 444.

2. Use the `ldapmodify` command to import the LDIF file into the directory.

3. Restart the server.

   For information on restarting the server, refer to “Starting and Stopping the Directory Server,” on page 29.

Configuring the Optional Parameters

You can configure the following optional parameters for the PTA plug-in:

- The maximum number of connections the PTA directory server can open simultaneously to the authenticating directory, represented by *maxconns* in the PTA syntax.The default value is `3`.
- The maximum number of bind requests the PTA directory server can send simultaneously to the authenticating directory server within a single connection. In the PTA syntax, this parameter is represented as *maxops*. The default is value is `5`.

- The time limit you want the PTA directory server to wait for a response from the authenticating directory server. In the PTA syntax, this parameter is represented as *timeout*. The default value is `300` seconds (five minutes).
- The version of the LDAP protocol you want the PTA directory server to use to connect to the authenticating directory server. In the PTA syntax, this parameter is represented as *ldver*. The default is `LDAPv3`.
- The time limit in seconds within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating directory server. The server will not close the connection unless a bind request is initiated and the server determines the timeout has been exceeded. If you do not specify this option, or if only one authenticating directory server is listed in the *authDS* parameter, no time limit will be enforced. If two or more hosts are listed, the default is 300 seconds (five minutes). In the PTA syntax, this parameter is represented as *connlifetime*.

NOTE Although these parameters are optional, if you specify one of them, you need to specify them all, even if you use the default values.

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
cn: Pass Through Authentication
changetype: add
add: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://authDS/subtree [maxconns,maxops,timeout,ldver,connlifetime]
```

   Make sure there is a space between the *subtree* parameter, and the optional parameters.

   For example, you could set the value of the `nsslapd-pluginarg0` attribute to:

   `"ldap://dirserver.siroe.com/o=NetscapeRoot 3,5,300,3,300"`

   In this example, each of the optional parameters is set to its default value.

2. Use the `ldapmodify` command to import the LDIF file into the directory.
3. Restart the server.

   For information on restarting the server, refer to "Starting and Stopping the Directory Server," on page 29.

PTA Plug-In Syntax Examples

This section contains the following examples of PTA plug-in syntax in the `dse.ldif` file:

- Specifying One Authenticating Directory Server and One Subtree
- Specifying Multiple Authenticating Directory Servers
- Specifying One Authenticating Directory Server and Multiple Subtrees
- Using Non-Default Parameter Values
- Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

Specifying One Authenticating Directory Server and One Subtree

This example configures the PTA plug-in to accept all defaults for the optional variables. This configuration causes the PTA directory server to connect to the authenticating directory server for all bind requests to the `o=NetscapeRoot` subtree. The hostname of the authenticating directory server is `config-dir.siroe.com`.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config-dir.siroe.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying Multiple Authenticating Directory Servers

If the connection between the PTA directory server and the authenticating directory server is broken or the connection cannot be opened, the PTA directory server sends the request to the next server specified (if any). You can specify as many authenticating directory servers as required.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config-dir.siroe.com/ou=NetscapeRoot
nsslapd-pluginarg1: ldap://config2-dir.siroe.com/ou=NetscapeRoot
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying One Authenticating Directory Server and Multiple Subtrees

The following example configures the PTA directory server to pass through bind requests for more than one subtree (using parameter defaults):

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config-dir.siroe.com/ou=NetscapeRoot
nsslapd-pluginarg1: ldap://config-dir.siroe.com/dc=siroe,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

Using Non-Default Parameter Values

This example uses a non-default value (10) only for the maximum number of connections parameter *maxconns*. Each of the other parameters is set to its default value. However, because one parameter is specified, all parameters must be defined explicitly in the syntax.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config-dir.siroe.com/ou=NetscapeRoot 10,5,300,3,300
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

If you want to specify a different pass-through subtree and optional parameter values for each authenticating directory server, you must specify more than one LDAP URL/optional parameters pair. Separate the LDAP URL/optional parameter pairs with a single space as follows.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Pass Through Authentication
nsslapd-pluginPath: /usr/iplanet/servers/lib/passthru-plugin.so
nsslapd-pluginInitfunc: passthruauth_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://config-dir.siroe.com/ou=NetscapeRoot 7,7,300,3,300
nsslapd-pluginarg1: ldap://config2-dir.siroe.com/dc=siroe,dc=com 7,7,300,3,300
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: passthruauth
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: pass through authentication plugin
```

Chapter 17

Using the Attribute Uniqueness Plug-In

The attribute uniqueness plug-in can be used to ensure that the attributes you specify always have unique values in the directory. You must create a new instance of the plug-in for every attribute for which you want to ensure unique values.

iPlanet Directory Server 5.0, provides a uid uniqueness plug-in that can be used to manage the uniqueness of the uid attribute.

This chapter describes the attribute uniqueness plug-in and the uid uniqueness plug-in in the following sections:

- Overview of the Attribute Uniqueness Plug-In
- Overview of the UID Uniqueness Plug-in
- Attribute Uniqueness Plug-In Syntax
- Creating an Instance of the Attribute Uniqueness Plug-In
- Configuring Attribute Uniqueness Plug-Ins
- Attribute Uniqueness Plug-In Syntax Examples
- Replication and the Attribute Uniqueness Plug-In

Overview of the Attribute Uniqueness Plug-In

The attribute uniqueness plug-in is a preoperation plug-in. This means that the plug-in checks all update operations before the server performs an LDAP operation. The plug-in determines whether the operation applies to an attribute and a suffix that you have configured it to monitor.

If an update operation applies to an attribute and suffix monitored by the plug-in, and it would cause two entries to have the same attribute value, then the server terminates the operation and returns an `LDAP_CONSTRAINT_VIOLATION` error to the client.

The attribute uniqueness plug-in performs a check on:

- A single attribute
- One or several subtrees

If you want to check uniqueness of several attributes, you must create a separate instance of the plug-in for each attribute you want to check.

You can also configure how the attribute uniqueness plug-in operates:

- It can check every entry in the subtrees you specify.

  For example, if your company, Siroe, hosts the directories for Company333 and Company999, when you add an entry such as `uid=jlittle,ou=people,o=Company333,dc=siroe,dc=com`, you need to enforce uniqueness only in the `o=Company333,dc=siroe,dc=com` subtree. You can do this by listing the DN of the subtree explicitly in the uid uniqueness plug-in configuration.

  This configuration option is explained in more detail in “Specifying a Suffix or Subtree,” on page 463.

- You can specify an object class pertaining to an entry in the DN of the updated entry, and perform the uniqueness check on all the entries beneath it.

  This option is useful in hosted environments. For example, when you add an entry such as `uid=jlittle,ou=people,o=Company333,dc=siroe,dc=com`, you can enforce uniqueness under the `o=Company333,dc=siroe,dc=com` subtree without listing this subtree explicitly in the configuration, but instead, by indicating a *marker object class*. If you specify that the marker object class is `organization`, the uniqueness check algorithm locates the entry in the DN that has this object class (`o=Company333`) and performs the check on all entries beneath it.

  Additionally, you can specify to check uniqueness only if the updated entry includes a specified object class. For example, you could specify to perform the check only if the updated entry includes `objectclass=inetorgperson`.

  This configuration option is explained in more detail in “Using the markerObjectClass and requiredObjectClass Keywords,” on page 464.

If you intend to use the attribute uniqueness plug-in in a replicated environment, refer to “Replication and the Attribute Uniqueness Plug-In,” on page 467.

Overview of the UID Uniqueness Plug-in

iPlanet Directory Server 5.0 provides an instance of the attribute uniqueness plug-in, the Uid Uniqueness plug-in. By default, the plug-in ensures that values given to the uid attribute are unique in the suffix you configured when installing the directory (the suffix corresponding to the `userRoot` database).

You can change the configuration to specify additional suffixes or subtrees, or by specifying to only perform the check under entries that contain a specified object class. The syntax and configuration of the uid uniqueness plug-in is the same as for any other attribute. For more information on the configuration changes you can make, see “Configuring Attribute Uniqueness Plug-Ins,” on page 461.

By default, the Uid Uniqueness plug-in is disabled because it affects the operation of multi-master replication. For information on using the attribute uniqueness plug-in in a replicated environment, refer to “Replication and the Attribute Uniqueness Plug-In,” on page 467.

Attribute Uniqueness Plug-In Syntax

Configuration information for the attribute uniqueness plug-in is specified in an entry under `cn=plugins,cn=config` entry. There are two possible syntaxes for `nsslapd-pluginarg` attributes. The differences are highlighted in the sections below.

Use the following syntax to perform the uniqueness check under a suffix or subtree:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: descriptive_plugin_namee
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute_name
nsslapd-pluginarg1: dn1
[ nsslapd-pluginarg2: dn2 ]
nsslapd-plugin-depends-on-type: database
```

```
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

Notes:

- You can specify any name you like in the `cn` attribute to name the plug-in. The name should be descriptive. This attribute does *not* contain the name of the attribute which is checked for uniqueness.
- You can specify only one attribute on which the uniqueness check will be performed.
- You can specify several DNs of suffixes or subtrees in which you want to perform the uniqueness check by incrementing the `nsslapd-pluginarg` attribute suffix by 1 each time.

The variable components of the attribute uniqueness plug-in syntax are described in Table 17-1.

Use the following syntax to specify to perform the uniqueness check below an entry containing a specified object class:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: descriptive_plugin_name
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: state
nsslapd-pluginarg0: attribute=attribute_name
nsslapd-pluginarg1: markerObjectClass=objectclass1
[ nsslapd-pluginarg2: requiredObjectClass=objectclass2 ]
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

Notes:

- You can specify any name you like in the `cn` attribute to name the plug-in. The name should be descriptive. This attribute does *not* contain the name of the attribute which is checked for uniqueness.

- You can specify only one attribute on which the uniqueness check will be performed.
- If the `nsslapd-pluginarg0` attribute begins with `attribute=` *attribute_name*, then the server expects that the `nsslapd-pluginarg1` attribute will include a `markerObjectClass`.

The variable components of the attribute uniqueness plug-in syntax are described in Table 17-1.

Table 17-1 Attribute Uniqueness Plug-In Variables

Variable	Definition
descriptive_plugin_name	Specifies the name of this instance of the attribute uniqueness plug-in. You do not have to include the name of the attribute for which you want to ensure uniqueness, but it is advisable. For example `cn= mail uniqueness,cn=plugins,cn=config`.
extension	File extension for the plug-in. The extension is always `.sl` on HP-UX, `.so` on all other UNIX platforms, and `.dll` on Windows NT.
state	Defines whether the plug-in is enabled or disabled. Acceptable values are **`on`** or **`off`** . See "Turning the Plug-in On or Off," on page 463 for more information.
attribute_name	The name of the attribute for which you want to ensure unique values. You can specify one attribute name only.
dn	The DN of the suffix or subtree in which you want to ensure attribute uniqueness. You can specify several suffixes or subtrees by incrementing the suffix of the `nsslapd-pluginarg` attribute by 1 for each additional suffix or subtree.
`attribute=`*attribute_name*	The name of the attribute for which you want to ensure unique values. You can specify one attribute name only.
`markerObjectClass=`*objectclass1*	Attribute uniqueness will be checked under the entry belonging to the DN of the updated entry that has the object class specified in the `markerObjectClass` keyword. Do not include a space before or after the equals sign.
`requiredObjectClass=`*objectclass2*	Optional. When you use the `markerObjectClass` keyword to specify the scope of the uniqueness check instead of a DN, you can optionally specify to perform the check only if the updated entry contains the objectclass specified in the `requiredObjectClass` keyword. Do not include a space before or after the equals sign.

Creating an Instance of the Attribute Uniqueness Plug-In

If you want to ensure that a particular attribute in your directory always has unique values, you must create an instance of the attribute uniqueness plug-in for the attribute you want to check. For example, if you want to ensure that every entry in your directory that includes a `mail` attribute has a unique value for that attribute, you must create a mail uniqueness plug-in.

To create an instance of the attribute uniqueness plug-in, you must modify the `dse.ldif` file to add an entry for the new plug-in under the `cn=plugins,cn=config` entry. The format of the new entry must conform to the syntax described in "Attribute Uniqueness Plug-In Syntax," on page 457.

For example, to instantiate the attribute uniqueness plug-in for the mail attribute, you would perform the following steps:

1. In the `dse.ldif` file, locate the entry for the uid uniqueness plug-in, `cn=uid uniqueness,cn=plugins,cn=config`.
2. Add the following lines for the mail uniqueness plug-in entry before or after the uid uniqueness plug-in entry:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.extension
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=siroe,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

3. Restart Directory Server.

In this example, the uniqueness check will be performed on every entry in the `dc=siroe,dc=com` entry that includes the `mail` attribute.

Configuring Attribute Uniqueness Plug-Ins

This section explains how to use Directory Server Console to view the plug-ins configured for your directory, and how to modify the configuration of the attribute uniqueness plug-ins.

Viewing Plug-In Configuration Information

From the Directory Server Console, you can display the configuration entry for attribute uniqueness plug-ins as follows:

1. On the Directory Server Console, click the Directory tab.
2. In the left navigation tree, expand the config folder, then the plugins folder.

   The list of plug-ins is displayed in the right navigation window. You should see the uid uniqueness plug-in and any other attribute uniqueness plug-ins that you created following the example given in "Creating an Instance of the Attribute Uniqueness Plug-In," on page 460.
3. In the right navigation window, double-click the plug-in entry you want to look at.

   The Property Editor is displayed. It contains a list of all the attributes and values for the plug-in.

Configuring Attribute Uniqueness Plug-Ins From the Directory Server Console

You can update plug-in configuration from Directory Server Console in several ways:

- From the Property Editor

  Display the Property Editor as explained in "Viewing Plug-In Configuration Information," on page 461, and edit the attribute value fields.
- From the Configuration tab

To modify an attribute uniqueness plug-in configuration from the Directory Server Console Configuration tab:

1. On the Directory Server Console, select the Configuration tab, then in the navigation tree, expand the Plugins folder, and select the attribute uniqueness plug-in that you want to modify.

   The configuration parameters for the plug-in are displayed in the left pane.

2. To turn the plug-in on or off, check or clear the Enable Plugin checkbox.

3. To add a suffix or subtree, click Add, and type a DN in the blank text field.

   If you do not want to add a DN, you can use the `markerObjectClass` keyword. If you use this syntax, you can click Add again to specify a requiredObjectClass as described in "Attribute Uniqueness Plug-In Syntax," on page 457.

NOTE You must not add an attribute name to the list. If you want to check the uniqueness of other attributes, you must create a new instance of the attribute uniqueness plug-in for the attribute you want to check. For information, refer to "Creating an Instance of the Attribute Uniqueness Plug-In," on page 460.

4. To delete an item from the list, place the cursor in the text field that you want to delete, and click Delete.

5. Click Save to save your changes.

Configuring Attribute Uniqueness Plug-Ins from the Command Line

This section provides information about configuring the plug-in from the command line. It covers the following tasks:

- Turning the Plug-in On or Off
- Specifying a Suffix or Subtree
- Using the markerObjectClass and requiredObjectClass Keywords

Turning the Plug-in On or Off

To turn the plug-in on from the command line, you must create an LDIF file that contains the following LDIF update statements:

```
dn: cn=descriptive_plugin_name,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginenabled
nsslapd-pluginenabled: on
```

Use the `ldapmodify` command to import the LDIF file into the directory. For detailed information on the `ldapmodify` command, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

To disable the plug-in, change the LDIF update statements to replace the `nsslapd-pluginenabled: on` statement, by the `nsslapd-pluginenabled: off` statement.

Whenever you enable or disable the plug-in, you must restart the server. For information on restarting the server, refer to “Starting and Stopping the Directory Server,” on page 29.

Specifying a Suffix or Subtree

You specify the suffix or subtrees under which you want the plug-in to ensure attribute uniqueness by using the `nsslapd-pluginarg` attribute in the entry defining the plug-in.

You can specify the subtree or subtrees by creating and LDIF file that contains update statements similar to those shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
changetype: add
nsslapd-pluginarg2: dc=iplanet,dc=sun,dc=com
nsslapd-pluginarg3: dc=iplanet,dc=netscape,dc=com
```

This example LDIF file will check uniqueness of the `mail` attribute under the subtrees `dc=siroe,dc=com`, and `dc=iplanet,dc=sun,dc=com`, and `dc=iplanet, dc=netscape.com`.

Use the `ldapmodify` command to import the LDIF file into the directory. For detailed information on the `ldapmodify` command, refer to *iPlanet Directory Server Configuration, Command, and File Reference*.

Whenever you make this type of configuration change, you must restart the server. For information on restarting the server, refer to “Starting and Stopping the Directory Server,” on page 29.

Using the markerObjectClass and requiredObjectClass Keywords

Instead of specifying a suffix or subtree in the configuration of an attribute uniqueness plug-in, you can specify to perform the check under the entry belonging to the DN of the updated entry that has the object class specified in the `markerObjectClass` keyword.

To specify to perform the uniqueness check under the entry in the DN of the updated entry that contains the organizational unit (ou) object class, you can create an LDIF file such as the one shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: markerObjectClass=ou
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

If you do not want the server to check every entry under the organizational unit entry, you can limit the scope by specifying to perform the check only if the updated entry contains a specified object class.

For example, if you check the uniqueness of the `mail` attribute, it is probably necessary to perform the check only when you add or modify entries that contain the `person` or `inetorgperson` object class.

You can restrict the scope of the check by using the `requiredObjectClass` keyword, as shown in the following example:

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: attribute=mail
```

```
nsslapd-pluginarg1: markerObjectClass=ou
nsslapd-pluginarg2: requiredObjectClass=person
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

You cannot repeat the `markerObjectClass` or `requiredObjectClass` keywords by incrementing the counter in the `nsslapd-pluginarg` attribute suffix.

NOTE The `nsslapd-pluginarg0` attribute always contains the name of the attribute for which you want to ensure uniqueness.

Attribute Uniqueness Plug-In Syntax Examples

This section contains examples of attribute uniqueness plug-in syntax in the `dse.ldif` file. All examples show the plug-in syntax as it appears on UNIX machines.

- Specifying One Attribute and One Subtree
- Specifying One Attribute and Multiple Subtrees

Specifying One Attribute and One Subtree

This example configures the plug-in to ensure the uniqueness of the `mail` attribute under the `dc=siroe,dc=com` subtree.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: dc=siroe,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

Specifying One Attribute and Multiple Subtrees

This example configures the plug-in to ensure the uniqueness of the `mail` attribute under the `l=Chicago,dc=siroe,dc=com` and `l=Boston,dc=siroe,dc=com` subtrees.

```
dn: cn=mail uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: mail uniqueness
nsslapd-pluginPath: /usr/iplanet/servers/lib/uid-plugin.so
nsslapd-pluginInitfunc: NSUniqueAttr_Init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: mail
nsslapd-pluginarg1: l=Chicago,dc=siroe,dc=com
nsslapd-pluginarg2: l=Boston,dc=siroe,dc=com
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: NSUniqueAttr
nsslapd-pluginVersion: 5.0
nsslapd-pluginVendor: Sun | Netscape Alliance
nsslapd-pluginDescription: Enforce unique attribute values
```

NOTE The `nsslapd-pluginarg0` attribute always contains the name of the attribute for which you want to ensure uniqueness. All other occurrences of the `nsslapd-pluginarg` (`nsslapd-pluginarg1` to `nsslapd-pluginarg`*x*) 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=siroe,dc=com` subtree and once under the `l=Boston,dc=siroe,dc=com` subtree. For example, the following would be allowed:

`mail=bjensen,l=Chicago,dc=siroe,dc=com`

`mail=bjensen,l=Boston,dc=siroe,dc=com`

If you want to ensure that only one instance of a value exists under both subtrees, you need to configure the plug-in to ensure uniqueness for the entire `dc=siroe,dc=com` subtree.

Replication and the Attribute Uniqueness Plug-In

When you use the attribute uniqueness plug-ins on Directory Servers involved in a replication agreement, you must think carefully about how to configure the plug-in on each server.

Consider the following cases:

- Simple replication with one supplier and one or several consumers
- Complex replication with multiple masters

Attribute uniqueness plug-ins do not perform any checking on attribute values when an update is performed as part of a replication operation.

Simple Replication Scenario

Because all modifications by client applications are performed on the supplier server, the attribute uniqueness plug-in should be enabled on the supplier. It is unnecessary to enable it on the consumer server.

Enabling the attribute uniqueness plug-in on the consumer will not prevent Directory Server from operating correctly, but is likely to cause a performance degradation.

Multi-Master Replication Scenario

In a multi-master replication scenario, the two masters act both as suppliers and consumers of the same replica. Because multi-master replication uses a loosely consistent replication model, enabling an attribute uniqueness plug-in on one of the servers is not sufficient to ensure that attribute values will be unique across both masters at any given time. Therefore, enabling an attribute uniqueness plug-in on one server can cause inconsistencies in the data held on each replica.

However, you can use an attribute uniqueness plug-in, providing all of the following conditions are met:

- The attribute on which you are performing the uniqueness check is a naming attribute
- The attribute uniqueness plug-in is enabled on both masters

When these conditions are met, attribute uniqueness conflicts are reported as naming conflicts at replication time. Naming conflicts require manual resolution. For information on how to resolve replication conflicts, refer to “Solving Common Replication Conflicts,” on page 312.

Appendixes

Appendix A, “LDAP Data Interchange Format”

Appendix B, “Finding Directory Entries”

Appendix C, “LDAP URLs”

Appendix D, “Internationalization”

Appendix A

LDAP Data Interchange Format

Directory Server uses the LDAP Data Interchange Format (LDIF) to describe a directory and directory entries in text format. LDIF is commonly used to build the initial directory database or to add large numbers of entries to the directory all at once. In addition, LDIF is also used to describe changes to directory entries. For this reason, most of Directory Server's command-line utilities rely on LDIF for either input or output.

Because LDIF is a text file format, you can create your LDIF files using virtually any language. All directory data is stored using the UTF-8 encoding of Unicode. Therefore, the LDIF files you create must also be UTF-8 encoded.

This chapter provides information about LDIF in the following sections:

- "LDIF File Format," on page 471
- "Specifying Directory Entries Using LDIF," on page 475
- "Defining Directories Using LDIF," on page 480
- "Storing Information in Multiple Languages," on page 482

For information on using LDIF to modify directory entries, see Chapter 2, "Creating Directory Entries.

LDIF File Format

LDIF consists of one or more directory entries separated by a blank line. Each LDIF entry consists of an optional entry ID, a required distinguished name, one or more object classes, and multiple attribute definitions.

The LDIF format is defined in RFC 2849 *The LDAP Data Interchange Format (LDIF)*. iPlanet Directory Server is compliant with this standard.

The basic form of a directory entry represented in LDIF is as follows:

```
dn: distinguished_name
objectClass: object_class
objectClass: object_class
...
attribute_type[;subtype]:attribute_value
attribute_type[;subtype]:attribute_value
...
```

You must supply the DN and at least one object class definition. In addition, you must include any attributes required by the object classes that you define for the entry. All other attributes and object classes are optional. You can specify object classes and attributes in any order. The space after the colon is also optional. For information on standard object classes and attributes, refer to *iPlanet Directory Server Schema Reference*.

Table A-1 describes the LDIF fields shown in the previous definition.

Table A-1 LDIF Fields

Field	Definition
[*id*]	Optional. A positive decimal number representing the entry ID. The database creation tools generate this ID for you. Never add or edit this value yourself.
dn: *distinguished_name*	Specifies the distinguished name for the entry. For a complete description of distinguished names, refer to the *iPlanet Directory Server Deployment Guide*.
objectClass: *object_class*	Specifies an object class to use with this entry. The object class identifies the types of attributes, or schema, allowed and required for the entry. See the *iPlanet Directory Server Schema Reference* for a list of standard object classes, and Chapter 9, “Extending the Directory Schema” for information on customizing the schema.
attribute_type	Specifies a descriptive attribute to use with the entry. The attribute should be defined either in the schema. See the *iPlanet Directory Server Schema Reference* for a list of standard attributes, and Chapter 9, “Extending the Directory Schema” for information on customizing the schema.

Table A-1 LDIF Fields *(Continued)*

Field	Definition
[*subtype*]	Optional. Specifies a subtype, either language, binary, or pronunciation. Use this tag to identify the language in which the corresponding attribute value is expressed, or whether the attribute value is binary or a pronunciation of an attribute value. For information on attribute subtypes, see "Adding an Attribute Subtype," on page 44. For a complete list of the supported subtypes tags, see Table D-2 on page 515.
attribute_value	Specifies the attribute value to be used with the attribute type.

The LDIF syntax for representing a change to an entry in the directory is different from the syntax described above. For information on using LDIF to modify directory entries, see Chapter 2, "Creating Directory Entries."

Continuing Lines in LDIF

When you specify LDIF, you can break and continue, or fold, a line by indenting the continued portion of the line by exactly one space. For example, the following two statements are identical:

```
dn: cn=Jake Lupinski,dc=siroe,dc=com
```

```
dn: cn=Jake Lup
 inski, dc=sir
 oe,dc=comcom
```

You are not required to break and continue LDIF lines. However, doing so may improve the readability of your LDIF file.

Representing Binary Data

You can represent binary data, such as a JPEG image, in LDIF using one of the following methods:

- The standard LDIF notation, the lesser than (<) symbol. For example:

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

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

```
version:1
```

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

```
prompt% ldapmodify -D userDN -w user_passwd
```

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

- Using base 64 encoding. You identify base 64 encoded data by using the :: symbol. For example:

```
jpegPhoto:: encoded_data
```

In addition to binary data, other values that must be base 64-encoded include:

- Any value that begins with a semicolon (;) or a space
- Any value that contains non-ASCII data, including new lines

Use the `ldif` command-line utility with the `-b` parameter to convert binary data to LDIF format:

```
ldif -b attribute_name
```

where *attribute_name* is the name of the attribute to which you are supplying the binary data. The binary data is read from standard input and the results are written to standard output. Thus, you should use redirection operators to select input and output files.

The `ldif` command-line utility will take any input and format it with the correct line continuation and appropriate attribute information. The `ldif` utility also assesses whether the input requires base 64 encoding. For example:

```
ldif -b jpegPhoto < mark.jpg > out.ldif
```

This example takes a binary file containing a JPEG-formatted image and converts it into LDIF format for the attribute named `jpegPhoto`. The output is saved to `out.ldif`.

The `-b` option specifies that the `ldif` utility should interpret the entire input as a single binary value. If `-b` is not present, each line is considered to be a separate input value.

Specifying Directory Entries Using LDIF

You can store many types of entries in your directory. This section concentrates on three of the most common types of entries used in a directory: organization, organizational unit, and organizational person entries.

The object classes defined for an entry are what indicate whether the entry represents an organization, an organizational unit, an organizational person, or some other type of entry. For a general discussion of the types of entries you canc reate in your directory, see the *iPlanet Directory Server Deployment Guide*. For a complete list of the object classes you can use by default in your directory and a list of the most commonly used attributes, see the *iPlanet Directory Server Schema Reference*.

Specifying Organization Entries

Directories often have at least one organization entry. Typically this is the first, or topmost entry in your directory. The organization entry often corresponds to the suffix set for your directory. For example, if your directory is defined to use a suffix of `dc=siroe,dc=com`, then you will probably have an organization entry in your directory named `dc=siroe,dc=com`.

The LDIF that you specify to define an organization entry should appear as follows:

```
dn: distinguished_name
objectClass: top
objectClass: organization
o: organization_name
list_of_optional_attributes
...
```

The following is a sample organization entry in LDIF format:

```
dn: dc=siroe,dc=com
objectclass: top
objectclass: organization
o: Siroe Corporation
description: Fictional company for example purposes
telephonenumber: 555-5555
```

The organization name in the following example uses a comma:

```
dn: o="Siroe Chile\\, S.A."
objectclass: top
objectclass: organization
o: “Siroe Chile\\, S.A.”
description: Fictional company for example purposes
telephonenumber: 555-5556
```

Each element of the LDIF-formatted organization entry is defined in Table A-2.

Table A-2 LDIF Elements in Organization Entries

LDIF Element	Description
`dn:` *distinguished_name*	Specifies the distinguished name for the entry. DNs are described in the *iPlanet Directory Server Deployment Guide*. A DN is required.
`objectClass: top`	Required. Specifies the `top` object class.
objectClass: organization	Specifies the `organization` object class. This line defines the entry as an organization. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.
o: *organization_name*	Attribute that specifies the organization’s name. If the organization name includes a comma, you must escape the comma by either a single backslash (on NT) or two backslashes (on UNIX) and the entire organization argument must be enclosed in quotation marks. For example, to set the suffix to Siroe Bolivia, S.A. you would enter `“o: Siroe Bolivia\\, S.A.”` on UNIX machines or `“o: Siroe Bolivia\, S.A.”` on Windows NT.
list_of_attributes	Specifies the list of optional attributes that you want to maintain for the entry. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Specifying Organizational Unit Entries

Organizational unit entries are often used to represent major branch points, or subdirectories, in your directory tree. They correspond to major, reasonably static entities within your enterprise, such as a subtree that contains people, or a subtree that contains groups. However, the organizational unit attribute that is contained in the entry may also represent a major organization within your enterprise, such as marketing or engineering.

There is usually more than one organizational unit, or branch point, within a directory tree. For information on how to design your directory tree, see the *iPlanet Directory Server Deployment Guide.*

The LDIF that you specify to define an organizational unit entry must appear as follows:

```
dn: distinguished_name
objectClass: top
objectClass: organizationalUnit
ou: organizational_unit_name
list_of_optional_attributes
...
```

The following is a sample organizational unit entry in LDIF format:

```
dn: ou=people, dc=siroe,dc=com
objectclass: top
objectclass: organizationalUnit
ou: people
description: Fictional organizational unit for example purposes
```

Table A-3 defines each element of the LDIF-formatted organizational unit entry.

Table A-3 LDIF Elements in Organizational Unit Entries

LDIF Element	Description
dn: *distinguished_name*	Specifies the distinguished name for the entry. A DN is required. If there is a comma in the DN, the comma must be escaped with a backslash (\). For example: `dn: ou=people,o=siroe bolivia\,S.A.`
objectClass: top	Required. Specifies the `top` object class.
objectClass: organizationalUnit	Specifies the `organizationalUnit` object class. This line defines the entry as an organizationalUnit. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Table A-3 LDIF Elements in Organizational Unit Entries *(Continued)*

LDIF Element	Description
`ou:` *organizational_unit_name*	Attribute that specifies the organizational unit's name.
list_of_attributes	Specifies the list of optional attributes that you want to maintain for the entry. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Specifying Organizational Person Entries

The majority of the entries in your directory represent organizational people.

In LDIF, the definition of an organizational person is as follows:

```
dn: distinguished_name
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: common_name
sn: surname
list_of_optional_attributes
```

The following is an example organizational person entry in LDIF format:

```
dn: uid=bjensen,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenname: Babs
uid: bjensen
ou: Marketing
ou: people
description: Fictional person for example purposes
telephonenumber: 555-5557
userpassword: {sha}dkfljlk34r2kljdsfk9
```

Table A-4 defines each aspect of the LDIF person entry.

Table A-4 LDIF Elements in Person Entries

LDIF Element	Description
dn: *distinguished_name*	Specifies the distinguished name for the entry. A DN is required. If there is a comma in the DN, the comma must be escaped with a backslash (\). For example, `dn:uid=bjensen,ou=people,o=siroe bolivia\,S.A.`
objectClass: top	Required. Specifies the `top` object class.
objectClass: person	Specifies the `person` object class. This object class specification should be included because many LDAP clients require it during search operations for a person or an organizational person.
objectClass: organizationalPerson	Specifies the `organizationalPerson` object class. This object class specification should be included because some LDAP clients require it during search operations for an organizational person.
objectClass: *inetOrgPerson*	Specifies the `inetOrgPerson` object class. The `inetOrgPerson` object class is recommended for the creation of an organizational person entry because this object class includes the widest range of attributes. The `uid` attribute is required by this object class, and entries that contain this object class are named based on the value of the `uid` attribute. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.
cn: *common_name*	Specifies the person's common name which is the full name commonly used by the person. For example, `cn: Bill Anderson`. At least one common name is required.
sn: *surname*	Specifies the person's surname, or last name. For example, `sn: Anderson`. A surname is required.
list_of_attributes	Specifies the list of optional attributes that you maintain for the entry. See the *iPlanet Directory Server Schema Reference* for a list of the attributes you can use with this object class.

Defining Directories Using LDIF

You can define the contents of an entire directory using LDIF. Using LDIF is an efficient method of directory creation when you have many entries to add to the directory.

To create a directory using LDIF, follow these steps:

1. Create an ASCII file containing the entries you want to add in LDIF format.

   Make sure each entry is separated from the next by an empty line. You should use just one line, and the first line of the file must not be blank or else the `ldapmodify` utility will exit. For more information, see “Specifying Directory Entries Using LDIF,” on page 475.

2. Begin each file with the topmost, or root, entry in the database.

   The root entry must represent the suffix or su-suffx contained by the database. For example, if your database has the suffix `dc=siroe,dc=com`, the first entry in your directory must be

   ```
   dn: dc=siroe,dc=com
   ```

   For information on suffixes, see the "Suffix" parameter described in the *iPlanet Directory Server Configuration, Command, and File Reference*.

3. Make sure that an entry representing a branch point in the LDIF file is placed before the entries that you want to create under that branch.

   For example, if you want to place an entry in a people and a group subtree, create the branch point for those subtrees before creating entries within those subtrees.

4. Create the directory from the LDIF file using one of the following methods:

   - Directory Server Console

     Use this method if you have a small database to import (less than 1000 entries). See “Performing an Import From the Console,” on page 134.

   - `ldif2db` command-line utility

     Use this method if you have a large database to import (more than 1,000 entries). See “Importing Using the ldif2db Command-Line Script,” on page 137.

   - `ldapmodify` command-line utility with the `-a` parameter

Use this method if you currently have a directory database, but you are adding a new subtree to the database. Unlike the other methods for creating the directory from an LDIF file, Directory Server must be running before you can add a subtree using `ldapmodify`. See “Adding and Modifying Entries Using ldapmodify,” on page 49.

LDIF File Example

The following example shows an LDIF file that contains one organization, two organizational units, and three organizational person entries:

```
dn: o=Siroe Corp,dc=siroe,dc=com
objectclass: top
objectclass: organization
o: Siroe Corp
description: Fictional organization for example purposes

dn: ou=People,o=Siroe Corp,dc=siroe,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People
description: Fictional organizational unit for example purposes
tel: 555-5559

dn: cn=June Rossi,ou=People,o=Siroe Corp,dc=siroe,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: June Rossi
sn: Rossi
givenName: June
mail: rossi@siroe.com
userPassword: {sha}KDIE3AL9DK
ou: Accounting
ou: people
telephoneNumber: 2616
roomNumber: 220

dn: cn=Marc Chambers,ou=People,o=Siroe Corp,dc=siroe,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Marc Chambers
sn: Chambers
givenName: Marc
mail: chambers@siroe.com
```

```
userPassword: {sha}jdl2alem87dlacz1
telephoneNumber: 2652
ou: Manufacturing
ou: People
roomNumber: 167

dn: cn=Robert Wong,ou=People,o=Siroe Corp,dc=siroe,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Robert Wong
cn: Bob Wong
sn: Wong
givenName: Robert
givenName: Bob
mail: bwong@siroe.com
userPassword: {sha}nn2msx761
telephoneNumber: 2881
roomNumber: 211
ou: Manufacturing
ou: people

dn: ou=Groups,o=Siroe Corp,dc=siroe,dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups
description: Fictional organizational unit for example purposes
```

Storing Information in Multiple Languages

If your directory contains a single language, you do not need to do anything special to add a new entry to the directory. However, if your organization is multinational, you may find it necessary to store information in multiple languages so that users in different locales can view directory information in their own language.

When information in your directory is represented in multiple languages, the server associates language tags with attribute values. When you add a new entry, you must provide attribute values used in the RDN (Relative Distinguished Name) without any language codes.

You can even store multiple languages within a single attribute. When you do, the attribute types are the same, but each value has a different language code.

For a list of the languages supported by Directory Server and their associated language tags, see “Identifying Supported Locales,” on page 513.

NOTE The language tag has no effect on how the string is stored within the directory. All object class and attribute strings are stored using UTF-8.

For example, suppose Siroe Corporation has offices in the United States and France and wants employees to be able to view directory information in their native language. When adding directory entries, the directory administrator chooses to provide attribute values in both English and French. When adding a directory entry for a new employee, Babs Jensen, the administrator creates the following LDIF entry:

```
dn: uid=bjensen,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
name: Babs Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
streetAddress: 1 University Street
streetAddress;lang-en: 1 University Street
streetAddress;lang-fr: 1 rue de l’Université
preferredLanguage: fr
```

Users accessing this directory entry with an LDAP client with the preferred language set to English will see the address `1 University Street`. Users accessing the directory with an LDAP client with the preferred language set to French will see the address `1 rue de l’Université`.

Appendix B

Finding Directory Entries

You can find entries in your directory using any LDAP client. Most clients provide some form of a search interface that allows you to easily search the directory and retrieve entry information.

NOTE You cannot search the directory unless the appropriate access control has been set in your directory. For information on setting access control in your directory, see “Managing Access Control,” on page 183.

This chapter covers the following topics:

- Finding Entries Using the Server Console
- LDAP Search Filters
- Using ldapsearch
- Searching an Internationalized Directory

Finding Entries Using the Server Console

Use the Directory tab of the Directory Server Console to browse the contents of the directory tree and search for specific entries in the directory.

1. Make sure the Directory Server is running.
2. Start Directory Server Console.

   See “Starting Directory Server Console,” on page 26 for specific instructions.

3. On Directory Server Console, select the Directory tab.

Depending on the DN you used to authenticate to the directory, this tab displays the contents of the directory that you have access permissions to view. You can browse through the contents of the tree or right-click an entry and select Search from the pop-up menu. See the online help for information on using this feature.

CAUTION Do not modify the contents of the `o=NetscapeRoot` suffix using the Directory tab unless instructed to do so by iPlanet Technical Support.

Using ldapsearch

You can use the `ldapsearch` command-line utility to locate and retrieve directory entries. This utility opens a connection to the specified server using the specified distinguished name and password, and locates entries based on a specified search filter. Search scopes can include a single entry, an entry's immediate subentries, or an entire tree or subtree.

Search results are returned in LDIF format.

This section contains information about the following topics:

- Using Special Characters
- ldapsearch Command-Line Format
- Commonly Used ldapsearch options
- ldapsearch Examples

Using Special Characters

When using the `ldapsearch` command-line utility, you may need to specify values that contain characters that have special meaning to the command-line interpreter (such as space [ ], asterisk [*], backslash [\], and so forth). When you specify special characters, enclose the value in quotation marks (“”). For example:

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

Depending on your command-line interpreter, use either single or double quotation marks for this purpose. Refer to your operating system documentation for more information.

ldapsearch Command-Line Format

When you use `ldapsearch`, you must enter the command using the following format:

`ldapsearch` [*optional_options*] [*optional_search_filter*] [*optional_list_of_attributes*]

where

- *optional_options* represents a series of command-line options. These must be specified before the search filter, if any.
- *optional_search_filter* represents an LDAP search filter as described in "LDAP Search Filters," on page 493. Do not specify a search filter if you are supplying search filters in a file using the `-f` option.
- *optional_list_of_attributes* represents a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see "Displaying Subsets of Attributes," on page 491. If you do not specify a list of attributes, the search returns values for all attributes permitted by the access control set in the directory (with the exception of operational attributes).

NOTE If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (*) in the list of attributes in the `ldapsearch` command.

Commonly Used ldapsearch options

The following lists the most commonly used ldapsearch command-line options. If you specify a value that contains a space [ ], the value should be surrounded by double quotation marks, for example, `-b "ou=groups, dc=siroe,dc=com"`.

`-b`	Specifies the starting point for the search. The value specified here must be a distinguished name that currently exists in the database. This option is optional if the `LDAP_BASEDN` environment variable has been set to a base DN.
	
	The value specified in this option should be provided in double quotation marks. For example:
	
	`-b "cn=Barbara Jensen, ou=Product Development, dc=siroe,dc=com".`
	
	If you want to search the root DSE entry, specify an empty string here. For example:
	`-b ""`

`-D`	Specifies the distinguished name with which to authenticate to the server. This option is optional if anonymous access is supported by your server. If specified, this value must be a DN recognized by the Directory Server, and it must also have the authority to search for the entries. For example: `-D "uid=bjensen, dc=siroe,dc=com".`
`-h`	Specifies the hostname or IP address of the machine on which the Directory Server is installed. If you do not specify a host, `ldapsearch` uses the localhost. For example, `-h mozilla`.
`-l`	Specifies the maximum number of seconds to wait for a search request to complete. Regardless of the value specified here, `ldapsearch` will never wait longer than is allowed by the server's `nsslapd-timelimit` attribute. For example, `-l 300`. The default value for the `nsslapd-timelimit` attribute is 3,600 seconds.
`-p`	Specifies the TCP port number that the Directory Server uses. For example, `-p 1049`. The default is 389. If `-Z` is used, the default is 636.

`-s`	Specifies the scope of the search. The scope can be one of the following: • `base`—Search only the entry specified in the `-b` option or defined by the `LDAP_BASEDN` environment variable. • `one`—Search only the immediate children of the entry specified in the `-b` option. Only the children are searched; the actual entry specified in the `-b` option is not searched. • `sub`—Search the entry specified in the `-b` option and all of its descendants. That is, perform a subtree search starting at the point identified in the `-b` option. This is the default.
`-w`	Specifies the password associated with the distinguished name that is specified in the `-D` option. If you do not specify this option, anonymous access is used. For example, `-w diner892`.
`-x`	Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server rather than on the client.
`-z`	Specifies the maximum number of entries to return in response to a search request. For example, `-z 1000`. Normally, regardless of the value specified here, `ldapsearch` never returns more entries than the number allowed by the server's `nsslapd-sizelimit` attribute. However, you can override this limitation by binding as the root DN when using this command-line argument. When you bind as the root DN, this option defaults to zero (0). The default value for the `nsslapd-sizelimit` attribute is 2,000 entries.

For detailed information on all `ldapsearch` utility options, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

ldapsearch Examples

In the next set of examples, suppose the following are true:

- You want to perform a search of all entries in the directory.

- You have configured your directory to support anonymous access for search and read. You do not have to specify any bind information in order to perform the search. For more information on anonymous access, see “Defining User Access - userdn Keyword,” on page 202.
- The server is located on hostname **`mozilla`**.
- The server uses port number **`389`**. Since this is the default port, you do not have to identify the port number on the search request.
- SSL is enabled for the server on port **`636`** (the default SSL port number).
- The suffix under which all data is stored is **`dc=siroe,dc=com`**.

Returning All Entries

Given the previous information, the following call will return all entries in the directory:

```
ldapsearch -h mozilla -b "dc=siroe,dc=com" -s sub "objectclass=*"
```

“`objectclass=*`” is a search filter that matches any entry in the directory.

Specifying Search Filters on the Command Line

You can specify a search filter directly on the command line. If you do this, be sure to enclose your filter in quotation marks (“filter”). Also, do not specify the `-f` option. For example:

```
ldapsearch -h mozilla -b "dc=siroe,dc=com" "cn=babs jensen"
```

Searching the Root DSE Entry

The root DSE, is a special entry that contains a list of all the suffixes supported by the local directory server. You can search this entry by supplying a search base of “”. You must also specify a search scope of `base` and a filter of `"objectclass=*"`. For example:

```
ldapsearch -h mozilla -b "" -s base "objectclass=*"
```

Searching the Schema Entry

iPlanet Directory Server stores all directory server schema in the special `cn=schema` entry. This entry contains information on every object class and attribute defined for your directory server.

You can examine the contents of this entry as follows:

```
ldapsearch -h mozilla -b "cn=schema" -s base "objectclass=*"
```

Using LDAP_BASEDN

To make searching easier, you can set your search base using the `LDAP_BASEDN` environment variable. Doing this allows you to skip specifying the search base with the `-b` option (for information on how to set environment variables, see the documentation for your operating system).

Typically, you set `LDAP_BASEDN` to your directory's suffix value. Since your directory suffix is equal to the root, or topmost, entry in your directory, this causes all searches to begin from your directory's root entry.

For example, suppose you have set `LDAP_BASEDN` to `dc=siroe,dc=com`. Then to search for `cn=babs jensen` in your directory use the following command-line call:

```
ldapsearch -h mozilla "cn=babs jensen"
```

In this example, the default scope of `sub` is used because the `-s` option was not used to specify the scope.

Displaying Subsets of Attributes

The `ldapsearch` command returns all search results in LDIF format. By default, `ldapsearch` returns the entry's distinguished name and all of the attributes that you are allowed to read (you can set up the directory access control such that you are allowed to read only a subset of the attributes on any given directory entry). Only operational attributes are not returned. If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command.

Suppose you do not want to see all of the attributes returned in the search results. You can limit the returned attributes to just a few specific attributes by specifying the ones you want on the command line immediately after the search filter. For example, to show the `cn` and `sn` attributes for every entry in the directory, use the following command-line call:

```
ldapsearch -h mozilla "objectclass=*" sn cn
```

This example assumes you set your search base with `LDAP_BASEDN`.

Specifying Search Filters Using a File

You can enter search filters into a file instead of entering them on the command line. When you do this, specify each search filter on a separate line in the file. The `ldapsearch` command runs each search in the order in which it appears in the file.

For example, if the file contains:

```
sn=Francis
givenname=Richard
```

then `ldapsearch` first finds all the entries with the surname Francis, and then all the entries with the givenname Richard. If an entry is found that matches both search criteria, then the entry is returned twice.

For example, suppose you specified the previous search filters in a file named `searchdb`, and you set your search base using `LDAP_BASEDN`. Then the following returns all the entries that match either search filter:

```
ldapsearch -h mozilla -f searchdb
```

You can limit the set of attributes returned here by specifying the attribute names that you want at the end of the search line. For example, the following `ldapsearch` command performs both searches, but returns only the DN and and the `givenname` and `sn` attributes of each entry:

```
ldapsearch -h mozilla -f searchdb sn givenname
```

Specifying DNs that Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, you must escape the comma with a backslash (\). For example, to find everyone in the Siroe Bolivia, S.A. subtree, use the following command:

```
ldapsearch -h mozilla -s base -b "o=Siroe Bolivia\, S.A.,dc=siroe,dc=com"
"objectclass=*"
```

Using Client Authentication When Searching

This example shows user `bjensen` searching the directory using client authentication:

```
ldapsearch -h mozilla -p 636 -b "dc=siroe,dc=com" -N
"bjensenscertname" -Z -W certdbpassword -P /home/bjensen/certdb/cert.db
"givenname=Richard"
```

LDAP Search Filters

Search filters select the entries to be returned for a search operation. They are most commonly used with the `ldapsearch` command-line utility. When you use `ldapsearch`, you can place multiple search filters in a file, with each filter on a separate line in the file, or you can specify a search filter directly on the command line.

For example, the following filter specifies a search for the common name Babs Jensen:

```
cn=babs jensen
```

This search filter returns all entries that contain the common name Babs Jensen. Searches for common name values are not case sensitive.

When the common name attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match this filter:

```
cn: babs jensen
```

```
cn;lang-fr: babs jensen
```

For a list of all the supported language tags, see Table D-1 on page 513.

Search Filter Syntax

The basic syntax of a search filter is:

attribute operator value

For example:

```
buildingname>=alpha
```

In this example, `buildingname` is the attribute, `>=` is the operator, and **`alpha`** is the value. You can also define filters that use different attributes combined together with Boolean operators.

Search filters are described in detail in the following sections:

- Using Attributes in Search Filters
- Using Operators in Search Filters
- Using Compound Search Filters
- Search Filter Examples

Using Attributes in Search Filters

When searching for an entry, you can specify attributes associated with that type of entry. For example, when you search for people entries, you can use the `cn` attribute to search for people with a specific common name.

Examples of attributes that people entries might include:

- `cn` (the person's common name)
- `sn` (the person's surname, or last name, or family name)
- `telephoneNumber` (the person's telephone number)
- `buildingName` (the name of the building in which the person resides)
- `l` (the locality where you can find the person)

For a listing of the attributes associated with types of entries, see the *iPlanet Directory Server Schema Reference*.

Using Operators in Search Filters

The operators that you can use in search filters are listed in Table B-1:

Table B-1 Search Filter Operators

Search type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, `cn=Bob Johnson`

Substring	=*string** *string*	Returns entries containing attributes containing the specified substring. For example,
		`cn=Bob*`
		`cn=*Johnson`
		`cn=*John*`
		`cn=B*John`
		(The asterisk (*) indicates zero (0) or more characters.)

Table B-1 Search Filter Operators *(Continued)*

Search type	Operator	Description
Greater than or equal to	>=	Returns entries containing attributes that are greater than or equal to the specified value. For example, `buildingname >= alpha`
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, `buildingname <= alpha`

Presence	=*	Returns entries containing one or more values for the specified attribute. For example,
		`cn=*`
		`telephonenumber=*`
		`manager=*`

Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example,
		`cn~=suret`
		`l~=san fransico`
		could return
		`cn=sarette`
		`l=san francisco`

NOTE In addition to these search filters, you can specify special filters to work with a preferred language collation order. For information on how to search a directory with international character sets, see “Searching an Internationalized Directory,” on page 497.

Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

(*Boolean-operator*(*filter*)(*filter*)(*filter*)...)

where *Boolean-operator* is any one of the Boolean operators listed in Table B-2.

Boolean operators can be combined and nested together to form complex expressions, such as:

(*Boolean-operator*(*filter*)((*Boolean-operator*(*filter*)(*filter*)))

The Boolean operators available for use with search filters include the following:

Table B-2 Search Filter Boolean Operators

Operator	Symbol	Description
`AND`	&	All specified filters must be true for the statement to be true. For example, `(&(filter)(filter)(filter)...)`
`OR`	|	At least one specified filter must be true for the statement to be true. For example, `(|(filter)(filter)(filter)...)`
`NOT`	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the `NOT` operator. For example, `(!(filter))`

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first
- All expressions from left to right

Search Filter Examples

The following filter searches for entries containing one or more values for the manager attribute. This is also known as a presence search:

```
manager=*
```

The following filter searches for entries containing the common name Ray Kultgen. This is also known as an equality search:

```
cn=Ray Kultgen
```

The following filter returns all entries that do not contain the common name Ray Kultgen:

```
(!(cn=Ray Kultgen))
```

The following filter returns all entries that contain a description attribute that contains the substring `X.500`:

```
description=*X.500*
```

The following filter returns all entries whose organizational unit is Marketing and whose description field does not contain the substring `X.500`:

```
(&(ou=Marketing)(!(description=*X.500*)))
```

The following filter returns all entries whose organizational unit is Marketing and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(|(manager=cn=Julie
Fulmer,ou=Marketing,dc=siroe,dc=com)(manager=cn=Cindy
Zwaska,ou=Marketing,dc=siroe,dc=com)))
```

The following filter returns all entries that do not represent a person:

```
(!(objectClass=person))
```

The following filter returns all entries that do not represent a person and whose common name is similar to `printer3b`:

```
(&(!(objectClass=person))(cn~=printer3b))
```

Searching an Internationalized Directory

When you perform search operations, you can request that the directory sort the results based on any language for which the server has a supporting collation order. For a listing of the collation orders supported by the directory, see "Identifying Supported Locales," on page 513.

NOTE When performing internationalized searches, you must perform an LDAP v3 search; therefore, do not specify the `-V2` option on the call to `ldapsearch`.

This section focuses on the matching rule filter portion of the `ldapsearch` syntax. For more information on general `ldapsearch` syntax, see “LDAP Search Filters,” on page 493. For information on searching internationalized directories using the Users and Groups portion of the iPlanet Console, refer to the online help or *Managing Servers with iPlanet Console*.

This section covers the following topics:

- Matching Rule Filter Syntax
- Supported Search Types
- International Search Examples

Matching Rule Filter Syntax

A matching rule provides special guidelines for how the directory compares strings during a search operation. In an international search, the matching rule tells the system what collation order and operator to use when performing the search operation. For example, a matching rule in an international search might tell the server to search for attribute values that come at or after llama in the Spanish collation order. The syntax of the matching rule filter is as follows:

attr`:`*matchingRule*`:=`*value*

where:

- *attr* is an attribute belonging to entries you’re searching for, such as `cn` or `mail`
- *matchingRule* is a string that identifies either the collation order or the collation order and a relational operator, depending on the format you prefer. For a discussion of matching rule formats, see “Matching Rule Formats,” on page 498.
- *value* is either the attribute value you want to search for or a relational operator plus the attribute value you want to search for.The syntax of the value portion of the filter depends on the matching rule format you use.

Matching Rule Formats

The matching rule portion of a search filter can be represented in several ways. The one you use is a matter of personal preference. The matching rule can be represented in the following ways:

- As the OID of the collation order for the locale on which you want to base your search.

- As the language tag associated with the collation order on which you want to base your search.
- As the OID of the collation order and a suffix that represents a relational operator.
- As the language tag associated with the collation order and a suffix that represents a relational operator.

The syntax for each of these options is discussed in the following sections:

- Using an OID for the Matching Rule
- Using a Language Tag for the Matching Rule
- Using an OID and Suffix for the Matching Rule
- Using a Language Tag and Suffix for the Matching Rule

Using an OID for the Matching Rule

Each locale supported by the directory server has an associated collation order OID. For a list of locales supported by the directory server and their associated OIDs, see Table D-1 on page 513.

You can use the collation order OID in the matching rule portion of the matching rule filter as follows:

attr`:`*OID*`:=(`*relational_operator value*`)`

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search for all `departmentNumber` attributes that are at or after N4709 in the Swedish collation order, use the following filter:

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

Using a Language Tag for the Matching Rule

Each locale supported by the directory server has an associated language tag. For a list of locales supported by the directory server and their associated language tags, see Table D-1 on page 513.

You can use the language tag in the matching rule portion of the matching rule filter as follows:

attr`:`*language-tag*`:=(`*relational_operator value*`)`

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search the directory for all description attributes with a value of `estudiante` using the Spanish collation order, use the following filter:

```
cn:es:== estudiante
```

Using an OID and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the OID in the matching rule portion of the filter. Combine the OID and suffix as follows:

attr`:`*OID+suffix*`:=`*value*

For example, to search for `businessCategory` attributes with the value `softwareprodukte` in the German collation order, use the following filter:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
```

The `.3` in the previous example is the equality suffix.

For a list of locales supported by the directory server and their associated OIDs, see Table D-1 on page 513. For a list of relational operators and their equivalent suffixes, see Table B-3 on page 501.

Using a Language Tag and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the language tag in the matching rule portion of the filter. Combine the language tag and suffix as follows:

attr`:`*language-tag+suffix*`:=`*value*

For example, to search for all surnames that come at or after `La Salle` in the French collation order, use the following filter:

```
sn:fr.4:=La Salle
```

For a list of locales supported by the directory server and their associated language tags, see Table D-1 on page 513. For a list of relational operators and their equivalent suffixes, see Table B-3 on page 501.

Using Wildcards in Matching Rule Filters

When performing a substring search using a matching rule filter, you can use the asterisk (*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter `l` and ends with the letter `n`, you would enter a `l*n` in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter `u`, you would enter a value of `u*` in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, you must escape the * with the designated escape sequence, \5c2a. For example, to search for all employees with `businessCategory` attribute values of `Siroe*Net product line`, enter the following value in the search filter:

```
Siroe\2a*Net product line
```

Supported Search Types

The directory server supports the following types of international searches:

- equality (=)
- substring (*)
- greater than (>)
- greater than or equal to (>=)
- less than (<)
- less than or equal to (<=)

Approximate, or phonetic, and presence searches are supported only in English.

As with a regular `ldapsearch` search operation, an international search uses operators to define the type of search. However, when invoking an international search, you can either use the standard operators (=, >=, >, <, <=) in the value portion of the search string, or you can use a special type of operator, called a suffix (not to be confused with the directory suffix), in the matching rule portion of the filter. Table B-3 summarizes each type of search, the operator, and the equivalent suffix.

Table B-3 Search Types, Operators, and Suffixes

Search Type	Operator	Suffix
Less than	`<`	`.1`
Less than or equal to	`<=`	`.2`
Equality	`=`	`.3`

Table B-3 Search Types, Operators, and Suffixes *(Continued)*

Search Type	Operator	Suffix
Greater than or equal to	`>=`	`.4`
Greater than	`>`	`.5`
Substring	`*`	`.6`

International Search Examples

The following sections show examples of how to perform international searches on directory data. Each example gives all the possible matching rule filter formats so that you can become familiar with the formats and select the one that works best for you.

Less Than Example

When you perform a locale-specific search using the less than operator (<) or suffix (.1), you search for all attribute values that come before the given attribute in a specific collation order.

For example, to search for all surnames that come before the surname Marquez in the Spanish collation order, you could use any of the following matching rule filters:

```
sn:2.16.840.1.113730.3.3.2.15.1:=< Marquez
sn:es:=< Marquez
sn:2.16.840.1.113730.3.3.2.15.1.1:=Marquez
sn:es.1:=Marquez
```

Less Than or Equal to Example

When you perform a locale-specific search using the less than or equal to operator (<=) or suffix (.2), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all room numbers that come at or before room number CZ422 in the Hungarian collation order, you could use any of the following matching rule filters:

```
roomNumber:2.16.840.1.113730.3.3.2.23.1:=<= CZ422
roomNumber:hu:=<= CZ422
roomNumber:2.16.840.1.113730.3.3.2.23.1.2:=CZ422
roomNumber:hu.2:=CZ422
```

Equality Example

When you perform a locale-specific search using the equal to operator (=) or suffix (.3), you search for all attribute values that match the given attribute in a specific collation order.

For example, to search for all `businessCategory` attributes with the value `softwareprodukte` in the German collation order, you could use any of the following matching rule filters:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1:== softwareprodukte
businessCategory:de:== softwareprodukte
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
businessCategory:de.3:=softwareprodukte
```

Greater Than or Equal to Example

When you perform a locale-specific search using the greater than or equal to operator (>=) or suffix (.4), you search for all attribute values that come at or after the given attribute in a specific collation order.

For example, to search for all localities that come at or after Québec in the French collation order, you could use any of the following matching rule filters:

```
locality:2.16.840.1.113730.3.3.2.18.1:=>= Québec
locality:fr:=>= Québec
locality:2.16.840.1.113730.3.3.2.18.1.4:=Québec
locality:fr.4:=Québec
```

Greater Than Example

When you perform a locale-specific search using the greater than operator (>) or suffix (.5), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all mail hosts that come after host `schranka4` in the Czechoslovakian collation order, you could use any of the following matching rule filters:

```
mailHost:2.16.840.1.113730.3.3.2.5.1:=> schranka4
mailHost:cs:=> schranka4
mailHost:2.16.840.1.113730.3.3.2.5.1.5:=schranka4
mailHost:cs.5:=schranka4
```

Substring Example

When you perform an international substring search, you search for all values that match the given pattern in the specified collation order.

For example, to search for all user IDs that end in `ming` in the Chinese collation order, you could use any of the following matching rule filters:

```
uid:2.16.840.1.113730.3.3.2.49.1:=* *ming
uid:zh:=* *ming
uid:2.16.840.1.113730.3.3.2.49.1.6:=* *ming
uid:zh.6:=* *ming
```

Appendix C

LDAP URLs

When you access the Directory Server using a web-based client such as Directory Server Gateway, you must provide an LDAP URL identifying the Directory Server you wish to access.

You also use LDAP URLs when managing Directory Server referrals or access control instructions.

This appendix contains the following sections:

- Components of an LDAP URL
- Escaping Unsafe Characters
- Examples of LDAP URLs

Components of an LDAP URL

LDAP URLs have the following syntax:

`ldap[s]://`*hostname*`:`*port*`/`*base_dn*`?`*attributes*`?`*scope*`?`*filter*

The `ldap://` protocol is used to connect to LDAP servers over unsecured connections, and the `ldaps://` protocol is used to connect to LDAP servers over SSL connections. Table C-1 lists the components of an LDAP URL.

Table C-1 LDAP URL Components

Component	**Description**
hostname	Name (or IP address in dotted format) of the LDAP server. For example: `ldap.siroe.com` or `192.202.185.90`

Table C-1 LDAP URL Components

Component	Description
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.
base_dn	Distinguished name (DN) of an entry in the directory. This DN identifies the entry that is the starting point of the search. If no base DN is specified, the search starts at the root of the directory tree.
attributes	The attributes to be returned. To specify more than one attribute, use commas to separate the attributes (for example, `"cn,mail,telephoneNumber"`). If no attributes are specified in the URL, all attributes are returned.
scope	The scope of the search, which can be one of these values: • `base` retrieves information only about the distinguished name (*base_dn*) specified in the URL. • `one` retrieves information about entries one level below the distinguished name (*base_dn*) specified in the URL. The base entry is not included in this scope. • `sub` retrieves information about entries at all levels below the distinguished name (*base_dn*) specified in the URL. The base entry is included in this scope. If no scope is specified, the server performs a `base` search.
filter	Search filter to apply to entries within the specified scope of the search. If no filter is specified, the server uses the filter `(objectClass=*)`.

The attributes, scope, and filter components are identified by their positions in the URL. If you do not want to specify any attributes, you still need to include the question marks delimiting that field.

For example, to specify a subtree search starting from `"dc=siroe,dc=com"` that returns all attributes for entries matching `"(sn=Jensen)"`, use the followingLDAP URL:

```
ldap://ldap.siroe.com/dc=siroe,dc=com??sub?(sn=Jensen)
```

The two consecutive question marks `??` indicate that no attributes have been specified. Since no specific attributes are identified in the URL, all attributes are returned in the search.

Escaping Unsafe Characters

Any “unsafe” characters in the URL need to be represented by a special sequence of characters. This is called escaping unsafe characters.

For example, a space is an unsafe character that must be represented as `%20` within the URL. Thus, the distinguished name `"o=siroe corporation"` must be encoded as `"o=siroe%20corporation"`.

The following table lists the characters that are considered unsafe within URLs and provides the associated escape characters to use in place of the unsafe character:

Unsafe Character	**Escape Characters**
space	%20
<	%3c
>	%3e
"	%22
#	%23
%	%25
{	%7b
}	%7d
|	%7c
\	%5c
^	%5e
~	%7e
[	%5b
]	%5d
‘	%60

Examples of LDAP URLs

- The following LDAP URL specifies a base search for the entry with the distinguished name `dc=siroe,dc=com`.

```
ldap://ldap.siroe.com/dc=siroe,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=siroe,dc=com`.
  - Because no filter is specified, the directory uses the default filter (`objectclass=*`).

- The following LDAP URL retrieves the `postalAddress` attribute of the entry with the DN `dc=siroe,dc=com`:

```
ldap://ldap.siroe.com/dc=siroe,dc=com?postalAddress
```

  - Because no search scope is specified, the search is restricted to the base entry `dc=siroe,dc=com`.
  - Because no filter is specified, the directory uses the default filter (`objectclass=*`).

- The following LDAP URL retrieves the `cn`, `mail`, and `telephoneNumber` attributes of the entry for Barbara Jensen:

```
ldap://ldap.siroe.com/cn=Barbara%20Jensen,dc=siroe,dc=com?cn,mail,telephoneNumber
```

  - Because no search scope is specified, the search is restricted to the base entry `cn=Barbara Jensen,dc=siroe,dc=com`.
  - Because no filter is specified, the directory uses the default filter `(objectclass=*)`.

- The following LDAP URL specifies a search for entries that have the surname `Jensen` and are at any level under `dc=siroe,dc=com`:

```
ldap://ldap.siroe.com/dc=siroe,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=siroe,dc=com` and entries at all levels under the base entry.

- The following LDAP URL specifies a search for the object class for all entries one level under `dc=siroe,dc=com`:

```
ldap://ldap.siroe.com/dc=siroe,dc=com?objectClass?one
```

  - Because the search scope is `one`, the search encompasses all entries one level under the base entry `dc=siroe,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

Directory Server allows you to store, manage, and search for entries and their associated attributes in a number of different languages. An internationalized directory can be an invaluable corporate resource, providing employees and business partners with immediate access to the information they need in the languages they can understand.

The directory supports all international characters set by default because directory data is stored in UTF-8. Further, Directory Server allows you to specify matching rules and collation orders based on language preferences in search operations.

NOTE You must use ASCII characters for attribute and object class names.

This appendix contains the following sections:

- About Locales
- Identifying Supported Locales
- Supported Language Subtypes

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, and/or custom expect data to be presented, including how data of a given language is interpreted and how data is to be sorted, or collated.

In addition, the locale information indicates what code page should be used to represent a given language. A code page is an internal table that the operating system uses to relate keyboard keys to character font screen displays.

More specifically, a locale specifies:

- Collation order

  The collation order provides language and cultural-specific information about how the characters of a given language are to be sorted. It identifies things like the sequence of the letters in the alphabet, how to compare letters with accents with letters without accents, and if there are any characters that can be ignored when comparing strings. The collation order also takes into account culture-specific information about a language, such as the direction in which the language is read (left to right, right to left, or up and down).

- Character type

  The character type distinguishes alphabetic characters from numeric or other characters. In addition, it defines the mapping of upper-case to lower-case letters. For example, in some languages, the pipe (|) character is considered punctuation while in others it is considered alphabetic.

- Monetary format

  The monetary format specifies the monetary symbol used by a specific region, whether the symbol goes before or after its value, and how monetary units are represented.

- Time/date format

  The time and date format indicates the customary formatting for times and dates in the region. The time and date format indicates whether dates are customarily represented in the `mm/dd/yy` (month, day, year), or `dd/mm/yy` (day, month, year) format and specifies what the days of the week and month are in a given language. For example, the date January 10, 1996 is represented as 10.leden 1996 in Czechoslovakian and 10 janvier 1996 in French.

Because a locale describes cultural, customary, and regional differences in addition to mechanical language differences, the directory data can both be translated into the specific languages understood by your users as well as be presented in a way that users in a given region expect.

Locale information is automatically copied to the `/usr/iplanet/servers/lib/nls/locale30` directory during Directory Server installation.

Identifying Supported Locales

When performing directory operations that require you to specify a locale, such as a search operation, you can use a language tag or a collation order object identifier (OID).

A language tag is a string that begins with the two-character lowercase language code that identifies the language (as defined in ISO standard 639). If necessary to distinguish regional differences in language, the language tag may also contain a country code, which is a two-character string (as defined in ISO standard 3166). The language code and country code are separated by a hyphen. For example, the language tag used to identify the British English locale is en-GB.

An object identifier (OID) is a decimal number used to uniquely identify an object, such as an attribute or object class. The OIDs you use when searching or indexing an internationalized directory identify specific collation orders supported by the Directory Server. For example, the OID `2.16.840.1.113730.3.3.2.17.1` identifies the Finnish collation order.

When performing an international search in the directory, use either the language tag or the OID to identify the collation order you want to use. However, when setting up an international index, you must use the OIDs. for more information on indexing, see Chapter 10, “Managing Indexes.”

The following table lists each locale supported by Directory Server and identifies the associated language tags and OIDs.

Table D-1 Supported Locales

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
Albanian	`sq`	`2.16.840.1.113730.3.3.2.44.1`
Arabic	`ar`	`2.16.840.1.113730.3.3.2.1.1`
Byelorussian	`be`	`2.16.840.1.113730.3.3.2.2.1`
Bulgarian	`bg`	`2.16.840.1.113730.3.3.2.3.1`
Catalan	`ca`	`2.16.840.1.113730.3.3.2.4.1`
Chinese (Simplified)	`zh`	`2.16.840.1.113730.3.3.2.49.1`
Chinese (Traditional)	`zh-TW`	`2.16.840.1.113730.3.3.2.50.1`
Croatian	`hr`	`2.16.840.1.113730.3.3.2.22.1`
Czechoslovakian	`cs`	`2.16.840.1.113730.3.3.2.5.1`
Danish	`da`	`2.16.840.1.113730.3.3.2.6.1`

Table D-1 Supported Locales *(Continued)*

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
English (US)	en or en-US	2.16.840.1.113730.3.3.2.11.1
Estonian	et	2.16.840.1.113730.3.3.2.16.1
Finnish	fi	2.16.840.1.113730.3.3.2.17.1
French	fr or fr-FR	2.16.840.1.113730.3.3.2.18.1
German	de	2.16.840.1.113730.3.3.2.7.1
Greek	el	2.16.840.1.113730.3.3.2.10.1
Hebrew	iw	2.16.840.1.113730.3.3.2.27.1
Hungarian	hu	2.16.840.1.113730.3.3.2.23.1
Icelandic	is	2.16.840.1.113730.3.3.2.24.1
Japanese	ja	2.16.840.1.113730.3.3.2.28.1
Korean	ko	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish	lv	2.16.840.1.113730.3.3.2.31.1
Lithuanian	lt	2.16.840.1.113730.3.3.2.30.1
Macedonian	mk	2.16.840.1.113730.3.3.2.32.1
Norwegian	no	2.16.840.1.113730.3.3.2.35.1
Polish	pl	2.16.840.1.113730.3.3.2.38.1
Romanian	ro	2.16.840.1.113730.3.3.2.39.1
Russian	ru	2.16.840.1.113730.3.3.2.40.1
Serbian (Cyrilic)	sr	2.16.840.1.113730.3.3.2.45.1
Serbian (Latin)	sh	2.16.840.1.113730.3.3.2.41.1
Slovakian	sk	2.16.840.1.113730.3.3.2.42.1
Slovenian	sl	2.16.840.1.113730.3.3.2.43.1
Spanish	es or es-ES	2.16.840.1.113730.3.3.2.15.1
Swedish	sv	2.16.840.1.113730.3.3.2.46.1
Turkish	tr	2.16.840.1.113730.3.3.2.47.1
Ukranian	uk	2.16.840.1.113730.3.3.2.48.1

Supported Language Subtypes

Language subtypes can be used by clients to determine specific values for which to search. For more information on using language subtypes, see “Adding an Attribute Subtype,” on page 44.

The following table contains the list of supported language subtypes.

Table D-2 Supported Language Subtypes

Language tag	**Language**
af	Afrikaans
be	Byelorussian
bg	Bulgarian
ca	Catalan
cs	Czechoslovakian
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fi	Finnish
fo	Faroese
fr	French
ga	Irish
gl	Galician
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ko	Korean

Table D-2 Supported Language Subtypes *(Continued)*

Language tag	Language
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sk	Slovakian
sl	Slovenian
sq	Albanian
sr	Serbian
sv	Swedish
tr	Turkish
uk	Ukrainian
zh	Chinese

Glossary

access control instruction *See ACI.*

ACI Access Control Instruction. An instruction that grants or denies permissions to entries in the directory.

access control list *See ACL.*

ACL Access control list. The mechanism for controlling access to your directory.

access rights In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, self-write, proxy and all.

account inactivation Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

All IDs Threshold A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

All IDs token A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index Allows for efficient approximate or "sounds-like" searches.

attribute Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list A list of required and optional attributes for a given entry type or object class.

authenticating directory server In pass-through authentication (PTA), the authenticating directory server is the directory server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the bind host.

authentication (1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

authentication certificate Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

base DN Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.

base distinguished name *See base DN.*

bind DN Distinguished name used to authenticate to Directory Server when performing an operation.

bind distinguished name *See bind DN.*

bind rule In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

branch entry An entry that represents the top of a subtree in the directory.

browser Software, such as Netscape Navigator, 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 Otherwise known as the virtual view index, speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branchpoint in the directory tree to improve display performance.

CA *See Certificate Authority*.

cascading replication In a cascading replication scenario, one server, often called the hub supplier acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a change log. 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 within 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.

change log A change log is record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers, or on other masters, in the case of multi-master replication.

character type Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

CIR *See consumer-initiated replication.*

class definition Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.

class of service *See CoS.*

classic CoS A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.

client *See LDAP client.*

code page An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.

collation order Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

consumer Server containing replicated directory trees or subtrees from a supplier server.

consumer-initiated replication Replication configuration where consumer servers pull directory data from supplier servers.

consumer server In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.

CoS A method for sharing attributes between entries in a way that is invisible to applications.

CoS definition entry Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry Contains a list of the shared attribute values.

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 file systems, 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 (DSGW) A collection of CGI forms that allows a browser to perform LDAP client functions, such as querying and accessing a Directory Server, from a web browser.

directory service A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

distinguished name String representation of an entry's name and location in an LDAP directory.

DIT *See directory tree.*

DM *See Directory Manager.*

DNS Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as `www.iPlanet.com`). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.

DNS alias A DNS alias is a hostname that the DNS server knows points to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as `www.[yourdomain].[domain]` might point to a real machine called `realthing.[yourdomain].[domain]` where the server currently exists.

DSGW *See Directory Server Gateway (DSGW).*

entry A group of lines in the LDIF file that contains information about an object.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

file extension The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename `index.html` the file extension is `html`.

file type The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter A constraint applied to a directory query that restricts the information returned.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

gateway *See Directory Server Gateway (DSGW).*

general access When granted, indicates that all authenticated users can access directory information.

hostname A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, `www.iPlanet.com` is the machine www in the subdomain iPlanet and com domain.

HTML Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Netscape Navigator how to display text, position graphics and form items, and display links to other pages.

HTTP Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.

HTTPD An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.

HTTP-NG The next generation of Hypertext Transfer Protocol.

HTTPS A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.

hub supplier In the context of replication, a server that holds a replica that is copied from a different server, and in turn replicates it to a third server. See also cascading replication.

index key Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index Speeds up searches for information in international directories.

International Standards Organization *See ISO.*

IP address Internet Protocol address. A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10).

ISO International Standards Organization

knowledge reference Pointers to directory information stored in different databases.

LDAP Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAPv3 Version 3 of the LDAP protocol, upon which Directory Server bases its schema format

LDAP client Software used to request and view LDAP entries from an LDAP Directory Server. See also *browser*.

LDAP Data Interchange Format *See LDAP Data Interchange Format.*

LDAP URL Provides the means of locating directory servers using DNS and then completing the query via LDAP. A sample LDAP URL is `ldap://ldap.iplanet.com`

LDBM database A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.

LDIF LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.

leaf entry An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol *See LDAP.*

locale Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

managed object A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role Allow you to create an explicit enumerated list of members.

management information base *See MIB.*

mapping tree A data structure that associates the names of suffixes (subtrees) with databases.

master agent *See SNMP master agent.*

matching rule Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5 A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data, that is unique with high probability, and is mathematically extremely hard to produce a piece of data that will produce the same message digest.

MD5 signature A message digest produced by the MD5 algorithm.

MIB Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace Management Information 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 change log for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor The server containing the database link that communicates with the remote server.

n + 1 directory problem The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions Multiple entries with the same distinguished name.

nested role Allow you to create roles that contain other roles.

network management application Network Management Station component that graphically displays information about SNMP managed devices (which device is up or down, which and how many error messages were received, etc.).

network management station *See NMS.*

NIS Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, file systems, and network parameters throughout a network of computers.

NMS Network Management Station. Powerful workstation with one or more network management applications installed.

ns-slapd iPlanet's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server. See also slapd.

object class Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

OID *See object identifier.*

operational attribute Operational attributes contain information used internally by the directory to keep track of modifications and subtree properties. They are not returned in response to a search unless explicitly requested.

parent access When granted, indicates that users have access to entries below their own in the directory tree, that is, 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 will pass through bind requests to the authenticating directory server from all clients whose DN is contained in this subtree.

password file A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as `/etc/passwd`, because of where it is kept.

password policy A set of rules that govern how passwords are used in a given directory.

permission In the context of access control, the permission states whether access to the directory information is granted or denied, and the level of access that is granted or denied. See access rights.

PDU Protocol Data Unit. Encoded messages which form the basis of data exchanges between SNMP devices.

pointer CoS A pointer CoS identifies the template entry using the template DN only.

presence index Allows you to search 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 Pass-through authentication. Mechanism by which one directory server consults another to check bind credentials.

PTA directory In pass-through authentication (PTA), the PTA directory server is the server that sends (passes through) bind requests it receives to the authenticating directory server.

PTA LDAP URL In pass-through authentication, the URL that defines the authenticating directory server, pass-through subtree(s) and optional parameters.

RAM Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.

rc.local A file on Unix machines that describes programs that are run when the machine starts. It is also called `/etc/rc.local` because of its location.

RDN Relative distinguished name. The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name.

referential integrity Mechanism that ensures that relationships between related entries are maintained within the directory.

referral (1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.

(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.

replica A database that participates in replication

read-only replica A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.

read-write replica A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.

relative distinguished name *See RDN.*

replication Act of copying directory trees or subtrees from supplier servers to consumer servers.

replication agreement Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the consumer servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

RFC Request For Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

role An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

root suffix The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

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, that is, 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 The server service is a process on Windows NT that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.

server root A directory on the server machine dedicated to holding the server program and configuration, maintenance, and information files.

Server Selector Interface that allows you select and configure servers using a browser.

service A background process on a Windows NT machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE Server Instance Entry.

Simple Network Management Protocol *See SNMP.*

single-master replication The most basic replication scenario in which two servers each hold a copy of the same read-write replicas to consumer servers. In a single-master replication scenario, the supplier server maintains a change log.

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 Simple Network Management Protocol. Used to monitor and manage application processes running on the servers, by exchanging data about network activity.

SNMP master agent Software that exchanges information between the various subagents and the NMS.

SNMP subagent Software that gathers information about the managed device and passes the information to the master agent.

SSL Secure Sockets Layer. A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP.

standard index Indexes that are 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 (also called root). The superuser has complete access privileges to all files on the machine.

supplier Server containing the master copy of directory trees or subtrees that are replicated to consumer servers.

supplier server In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication Replication configuration where supplier servers replicate directory data to consumer servers.

symmetric encryption Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index Cannot be deleted or modified as it is essential to Directory Server operations.

target In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry The entries within the scope of a CoS.

TCP/IP Transmission Control Protocol/Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.

template entry *See CoS template entry.*

time / date format Indicates the customary formatting for times and dates in a specific region.

TLS Transport Layer Security. The new standard for secure socket layers, a public key based protocol.

topology The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security *See TLS.*

uid A unique number associated with each user on a Unix system.

URL Uniform Resource Locator. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is `[protocol]://[machine:port]/[document]`. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

virtual list view index Otherwise known as a browsing index, speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branchpoint in the directory tree to improve display performance.

X.500 standard The set of ISO / ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementations.

Index

A

access control
- ACI attribute 184
- ACI syntax 188
- allowing or denying access 197
- and replication 250
- and schema checking 192
- anonymous access 202, 216, 225
- bind rules 200
  - access at specific time or day 214
  - access based on value matching 207
  - general access 203
  - user and group access 202
- Boolean bind rules 217
- compatibility with earlier versions 251
- creating from console 218
- dynamic targets 203
- from specific domain 213
- from specific IP address 213
- logging information 251
- overview 183, 184
- permissions 196
- placement of ACIs 185
- rights 197
- roles 165
- SASL authentication 216
- simple authentication 216
- SSL authentication
- structure of ACIs
- target DN containing comma 242
- target DN containing comma and 191
- targeting 189
- targeting attribute values 194
- targeting attributes 192
- targeting entries 190
- targeting using filters 193
- using the Access Control Editor 218
- value matching 207

Access Control Editor
- displaying 219
- viewing current ACIs 221

access control instruction (ACI). See ACI

access log
- configuring 380
- manually rotating 384
- turning off 380
- turning on 380
- viewing 379

account inactivation 263
- from command line 264
- from console 264

account lockout 260
- configuration
  - attributes 261
- configuring 260
  - using command line 260
  - using console 260
- disabling 260
- enabling 260
- lockout duration 260
- password failure counter 260

ACI
- assessment
- attribute 185
- authmethod keyword 216
- bind rules 188, 200

cascading chaining 118
creating from console 222
dayofweek keyword 215
deleting from console 224
dns keyword 213
editing from console 223
evaluation 185
examples of use 224
groupdn keyword 206
inheritance 210
ip keyword 213
local evaluation
  cascading chaining 119
name 188
permissions 188, 196
precedence rule 185
proxy rights example 242
replication 250
rights 197
roledn keyword 207
structure
syntax 188
targattrfilters keyword 194
target 188
target DN containing comma 242
target DN containing comma and 191
target keywords 190
target overview 189
targetattr keyword 192
targetfilter keyword 193
userattr and parent 210
userattr keyword 208
using macro ACIs 244
value-based 194
viewing current 221, 243
wildcard in target 191
wildcards 204

ACI attribute
  default index for 331
  overview 184

ACI placement 185

ACL. See ACI

activating accounts
  from command line 266
  from console 265

add right 197

adding directory entries 50

Administration Server
  master agents and 400

agents
  master agent 400
    Unix 400
    Windows NT 400
  subagent 400
    configuring 408
    enabling 408
    starting and stopping on Unix 407
    starting and stopping on Windows NT 407

AIX SNMP daemon 406

algorithm
  search 332

algorithms
  metaphone phonetic algorithm 333

all keyword 203

allowed attributes
  creating 324
  deleting 324, 325
  editing in object class 325

allowing access 197

anonymous access 216
  example 205, 225
  overview 202

anyone keyword 202

approximate index 328
  query string codes 333

approximate search 495

attribute
  ACI 184, 185
  adding 59
  adding multiple values 44
  adding to entry 43
  creating 324
  defining 320
  deleting 59, 321
  deleting from object class 324, 325
  deleting using LDIF update statements 62
  multi-valued 320, 321
  nsslapd-schemacheck 326
  OID 320, 321
  passwordChange 256
  passwordExp 256

passwordInHistory 258
passwordMustChange 256
passwordStorageScheme 259
ref 128
removing a value 44
roles 163
searching for 494
standard 317, 318
syntax 320, 321
targeting 192
user-defined 318
viewing 318
attribute subtypes 44
adding 46
binary 45
language 45
pronunciation 45
attribute type field (LDIF) 472
attribute uniqueness plug-in. See unique attribute plug-in
attribute value field (LDIF) 473
attribute values
adding 59
deleting 62
modifying 61
replacing 59
syntax 320, 321
attributes values
targeting 194
audit log
configuring 383
disabling 383
enabling 383
viewing 382
authentication
access control and 216
bind DN 28
certificate-based 371
LDAP URLs and 509
over SSL 368
authmethod keyword 216

B

backing up data 144
all 145
db2bak 145
dse.ldif 147
bak2db script 148
bak2db.pl perl script 149
base 64 encoding 473
base DN, ldapsearch and 491
binary data, LDIF and 473
binary subtype 45
bind credentials
for database links 97
bind DN
accessing the server 28
resource limits based on 266
viewing current 29
bind rules
access at specific time or day 214
access based on authentication method 216
LDIF example 216
access based on value matching
overview 207
ACI syntax 188
all keyword 203
anonymous access 202
example 205, 225
LDIF example 205
anyone keyword 202
authmethod keyword 216
Boolean 217
dayofweek keyword 215
dns keyword 213
general access 203
example 205
group access 206
group access example 232
groupdn keyword 206
ip keyword 213
LDAP URLs 203
LDIF keywords 201
overview 200
parent keyword 203
role access 207
roledn keyword 207

self keyword 203
timeofday keyword 214
user access
LDIF example 204
parent 203
self 203
user access example 227
userattr keyword 208
userdn keyword 202
Boolean bind rules
example 217
overview 217
Boolean operators, in search filters 496

C

cache hit ratio 393
cascading chaining
client ACIs 119
configuration attributes 120
configuring defaults 115
configuring from command line 117
configuring from console 116
example 121
local ACI evaluation 119
loop detection 120
overview 113
proxy admin user ACI 118
proxy authorization 118
cascading replication
initializing the replicas 290
introduction 276
setting up 285
certificate
mapping to a DN 372
password 33
certificate database
password 362
certificate-based authentication 371
setting up 371
chaining
cascading 113
component operations, from console 91
component operations,from command line 91
overview 87
using SSL 104
change log 271
deleting 297
using with referential integrity 66
change operations 55
add 59
delete 59
replace 59
change type
add 55
delete 63
LDIF 54
modify 59
character type 512
ciphers 370
DES 370
FIPS DES 370
FIPS Triple DES 370
list of 369
none-MD5 370
overview 369
RC2 370
RC4 369
selecting 369
class of service (CoS) 166
access control 182
classic
example 170
overview 170
cosPriority attribute 177
creating 171
definition entry 174, 175
editing 173
indirect
example 169
overview 169
pointer
example 169
overview 169
qualifiers 175
template entry
creating 177
overview 168
classic CoS
example 170

overview 170
client
using to find entries 485
client authentication
over SSL 372
code page 511
collation order
international indexing 337
overview 512
search filters and 497
command line
providing input from 47
command-line scripts
db2bak 145
command-line utilities
certificate-based authentication and 371
ldapdelete 52
ldapmodify 49
ldapsearch 493
ldif 474
ldif2db 341, 345, 349, 352
start-slapd 30
stop-slapd 30
commas, in DNs 54, 486
ACI targets and 191, 242
specifying LDIF entries with 477, 479
specifying suffix with 476
using ldapsearch with 492
compare right 197
compatibility
ACIs 251
replication 272
compound search filters 496
configuration attributes
account lockout 261
cascading chaining 120
password policy 256
suffix 76
connections
monitoring 387, 390, 391
viewing number of 386
console
starting 26
consumer initialization 297
manual consumer creation 299
online consumer creation 298
consumer server 270
continued lines
in LDIF 473
in LDIF update statements 55
CoS definition entry
attributes 175
object classes 174
CoS template entry 168
creating 177
CoS. See class of service.
cosPriority attribute 177
counter, password failures 260
country code 513
creating a database
from the command line 83
from the console 83
creating the directory 480
custom distribution function
adding to suffix 85
custom distribution logic
adding databases 84
adding to suffix 85

D

dash, in change operation 55
data consistency
using referential integrity 64
database
and associated suffix 71
backing up
db2bak 145
backup 144
backup files 145
backup from console 145
creating from command line 83
creating from console 83
creating multiple 84
creating using LDIF 480
deleting 87
export 140
db2ldif 143

export from console 142
import 133
ldif2db 137
ldif2db.pl 138
ldif2ldap 140
initialization 136
making read-only 86
monitoring from command-line 395
monitoring from server console 391
overview 81
read-only mode 151
replication 271
restore 144
restoring 413
bak2db 148
bak2db.pl 149
restoring from console 147
selecting for monitoring 391
viewing backend information 391
database link
cascading
configuring defaults 115
configuring from command line 117
configuring from console 116
overview 113
chaining with SSL 104
configuration 94
configuration attributes 100
configuration example 100, 102
configuring bind credentials 97
configuring failover servers 99
configuring LDAP URL 99
configuring suffix 96
creating from command line 96
creating from console 94
deleting 106
maintaining remote server info 105
overview 87
database server parameters
read-only 392
database transaction logging
described 413
durable transactions 415
log file location 414
databases
in directory server 71
date format 512
dayofweek keyword 215
db2bak script 145
db2bak utility 145
db2ldif utility 143
default referrals
setting 126
setting from console 126
settings from command line 126
defining
access control policy 218
attributes 320
object classes 323
definition entry. See CoS definition entry.
delete right 197
deleting
ACI 224
attribute values 62
attributes 59, 62
attributes from an object class 324, 325
database link 106
entries 63
multiple attributes 59
object classes 325
deleting directory entries 52
denying access 197
precedence rule 185
DES cipher 370, 371
directory creation 480
directory entries
adding using LDIF 49
creating 39, 50
deleting 46, 52
managing from command line 47
managing from console 37
modifying 41, 51
moving 58
renaming 58
Directory Manager
attribute 27
configuring 27
privileges 27
directory server 384
attributes 27
basic administration 25

binding to 28
changing bind DN 28
configuration 31
controlling access 183
creating a root entry 38, 48
creating content 133
creating entries 39, 50
data 133
databases 71
deleting entries 46, 52
importing data 133
international character sets 511
login 28
management console 26
managing entries 37
MIB 402
modifying entries 41, 51
monitoring 377, 384
monitoring from command line 389
monitoring with SNMP 399
overview 26
performance counters 384, 391
plug-ins 419
SNMP traps 401
starting and stopping 29
starting the console 26
suffixes 71
supported languages 513

directory trees
finding entries in 486

disabling suffixes 80

disk space
access log and 380
log files and 384

distribution function 84, 85

dn field (LDIF) 472

dn.db2 file 331

dn2id.db2 file 331

dns keyword 213

dse.ldif
PTA plugin 446

dse.ldif file
backing up 147
PTA syntax 446
restoring 151

durable transactions 415

dynamic groups 155
creating 155
modifying 155

E

end of file marker 47

entries
adding an object class 42
adding attributes 43
adding using LDIF 49
adding using LDIF update statements 55
cache hit ratio 393
creating 39, 50
using LDIF 475
deleting 46, 52
using ldapdelete 52
deleting using LDIF update statements 63
distribution 82
finding 486
managing 37
managing from command line 47
managing from console 37
modifying 41, 49, 51
using ldapmodify 49
using LDIF update statements 59
moving 58
order of creation 48
order of deletion 52, 63
removing an object class 43
renaming 58
root 480
targeting 190

entry cache hit ratio 393

entry distribution 82

environment variables
LDAP_BASEDN 491

EOF marker 47

equality index 328

equality search 494
example 497
international example 503

error log
access control information 251

configuring 381
manually rotating 384
turning off 381
turning on 381
viewing 381
example
cascading chaining 121
exporting data 140
db2ldif 143
using console 142
extending the directory schema 317

F

failover servers
for database links 99
files
access log 379
database backup 145
dn.db2 331
dn2id.db2 331
EOF marker 47
error log 381
id2children.db2 331
id2entry.db2 331
filtered role
creating 159
example 164
finding
attributes 494
entries 486
FIPS DES cipher 370
FIPS Triple DES cipher 370
format, LDIF 471

G

general access
example 205
overview 203
glossary of terms 517
greater than or equal to search
international example 503
overview 495
groupdn keyword 206
LDIF examples 206
groupdnattr keyword 208
groups
access control 202
access control example 232
access to directory 206
dynamic 155
creating 155
modifying 155
overview 153
static 154
creating 154
modifying 155

H

hub supplier 270
configuration 295

I

id field (LDIF) 472
id2children.db2 file 331
id2entry.db2 file 331
importing data 133
from console 134
ldif2ldap 140
using ldif2db 137
using ldif2db.pl 138
inactivating accounts 263
inactivating roles 161
index types 328
approximate index 328
equality index 328
presence index 328
indexes
creating dynamically 338

dynamic changes to 338
presence 331
indexing 328
creating indexes from console 337
system indexes 331
indirect CoS
example 169
overview 169
initializing databases 136
initializing replicas
cascading replication 290
multi-master replication 285
international character sets 511
international indexing
collation order 337
international searches 497
equality 503
examples 502
greater than 503
greater than or equal to 503
less than 502
less than or equal to 502
matching rule filter syntax 498
substring 503
using OIDs 499
internationalization
character type 512
collation order 512
country code 513
date format 512
language tag 513
locales and 511
location of files 512
matching rule filters 498
modifying entries 64
monetary format 512
object identifiers and 513
of LDIF files 482
search filters and 497
supported locales 513
time format 512
ip keyword 213

J

jpeg images 473

L

language code
in LDIF entries 482
list of supported 513
language subtype 45
language support
language tag 513
searching and 497
specifying using locales 513
language tags
described 513
in international searches 499
in LDIF update statements 64
LDAP clients
authentication over SSL 373
certificate-based authentication and 371
monitoring database with 395
monitoring server with 389
schema and 317
using to find entries 485
LDAP Data Interchange Format, see LDIF 54
LDAP search filters
DNs with commas and 492
in targets 193
example 240
examples 194
LDAP URLs
components of 505
examples 508
for database links 99
in access control 203
security and 509
syntax 505
LDAP_BASEDN environment variable 491
ldapdelete utility 49
deleting entries 52
DNs with commas and 54
example 53
ldapmodify utility 49

attributes with language tags 64
creating a root entry 48
creating entries 50
DNs with commas and 54
example 50, 51
example of use 50, 51
modifying entries 49, 51
schema checking and 50
vs. ldapdelete 49

ldapsearch utility
base DN and 491
commonly used options 487
DNs with commas and 486, 492
example of use 489
format 487
international searches 497
limiting attributes returned 491
search filters 493
specifying files 491
using 486

LDIF
access control keywords
groupdnattr 208
userattr 208
adding entries 49
binary data 473
change type 54
entry format 471
organization 475
organizational person 478
organizational unit 477
example 481
internationalization and 482
line continuation 473
Server Console and 49
specifying entries
organization 475
organizational person 478
organizational unit 477
update statements 54
using to create directory 480

LDIF entries
binary data in 473
commas in 476, 477, 479
creating 475
organizational person 478
organizational units 477
organizations 475
internationalization and 482

LDIF files
continued lines 473
creating directory using 480
creating multiple entries 49
example 481
importing from Server Console 49
internationalization and 482

LDIF format 471

LDIF update statements 54
adding attributes 59
adding entries 55
continued lines 55
deleting attribute values 62
deleting attributes 62
deleting entries 63
modifying attribute values 61
modifying entries 59
syntax 55

ldif utility
converting binary data to LDIF 474

ldif2db utility 137
options 341, 345, 349, 352

ldif2db.pl perl script 138

ldif2ldap utility 140

legacy consumer
configuration 307

legacy replication plug-in
overview 273

less than or equal to search
international example 502
syntax 495

less than search
international example 502
syntax 495

locales
defined 511
location of files 512
supported 513

locked accounts 260

lockout duration 260

log files 377
access log 379
audit log 382

database transaction 413
deletion policy 378
error log 380
location of 384
manual rotation 384
manually rotating 384
rotation policy 378

login identity
changing 28
viewing 29

loop detection
cascading chaining 120

M

macro ACIs
example 244
overview 244
syntax 247

managed device
managed device-initiated communication 401
overview 399

managed object 400

managed role
creating 158
example 163

manually rotating log files 384

markerObjectClass keyword 464

master agent
overview 400
Unix 400
Windows NT 400

matchingRule format 498
using language tag 499
using language tag and suffix 500
using OID 499
using OID and suffix 500

MD5 message authentication 370

metaphone phonetic algorithm 333

MIB
directory server 402
location of 402
netscape-ldap.mib 402
entries table 404
operations table 402

modifying
attribute values 61
entries 59
international entries 64

modifying directory entries 51

monetary format 512

monitoring
database from command-line 395
database from server console 391
directory server 377
from console 384
log files 377
replication status 311
threads 387
with SNMP 399

monitoring from console 384

moving entries 58

multi-master replication
initializing replicas 285
introduction 274
setting up 280

multiple search filters 496

N

naming conflicts
in replication 313

nested role
creating 160
example 165

netscape-ldap.mib 402
entries table 404
location of 402
operations table 402

network management station (NMS)
NMS-initiated communication 401

nsslapd-lookthroughlimit attribute
role in searching algorithm 333

nsslapd-schemacheck attribute 326

nsslapd-sizelimit attribute
role in searching algorithm 333

nsslapd-timelimit attribute
    role in searching algorithm 333

O

object class
    adding to an entry 42
    creating 323
    deleting 325
    editing 324
    name 323
    OID 323
    parent object 324
    referral 128
    removing from an entry 43
    roles 163
    standard 317, 322
    user-defined 322
    viewing 322
object identifier (OID) 513
    attribute 320, 321
    in matchingRule 499
    object class 323
objectClass field (LDIF) 472
OID, *See* object identifier
operations table 402
operations, defined 386
operators
    Boolean 496
    international searches and 501
    search filters and 494
    suffix 501
optional attributes
    creating 324
    deleting 324, 325
    editing 325
    editing in object class 325
organization, specifying entries for 475
organizational person, specifying entries for 478
organizational unit, specifying entries for 477

P

parent access 203
parent keyword 203
parent object 324
pass-through authentication (PTA). See PTA plug-in
password file
    SSL certificate 33
password policy
    account lockout 260
    attributes 256
    configuring 254
        using command line 255
        using console 254
    lockout duration 260
    managing 253
    password failure counter 260
    replication 262
passwordChange attribute 256
passwordExp attribute 256
passwordInHistory attribute 258
passwordMustChange attribute 256
passwords
    account lockout 260
    certificate 33
    failure counter 260
    lockout duration 260
    setting 259
passwordStorageScheme attribute 259
PDUs 400
performance counters 391
    monitoring the server with 384
performance tuning
    database 410
    server 409
permissions
    ACI syntax 188
    allowing or denying access 197
    assigning rights 197
    overview 196
    precedence rule 185
plug-in functions 419
plug-ins
    7-bit check plug-in 420
    ACL plug-in 421

ACL preoperation plug-in 421
binary syntax plug-in 422
Boolean syntax plug-in 422
case exact string syntax plug-in 423
case ignore string syntax plug-in 423
chaining database plug-in 424
Class of Service plug-in 424
CLEAR password storage plug-in 430
country String Syntax Plug-In 425
CRYPT password storage plug-in 430
disabling 439
distinguished name syntax plug-in 425
enabling 439
generalized time syntax plug-in 426
integer syntax plug-in 427
internationalization plug-in 427
ldbm database plug-in 428
legacy replication plug-in 428
multimaster replication plug-in 429
NS-MTA-MD5 password storage plug-in 431
octet string syntax plug-in 429
postal address string syntax plug-in 433
PTA plug-in 433
reference 419
referential integrity plug-in 434
retro change log plug-in 435
roles plug-in 435
SHA password storage plug-in 432
SSHA password storage plug-in 432
telephone syntax plug-in 436
uid uniqueness plug-in 437
URI plug-in 439

pointer CoS
example 169
overview 169

port number
directory server configuration 31
for SSL communications 31

precedence rule
ACI 185

preferences
security 369

presence index 328
defaults 331

presence search
example 496
syntax 495

pronunciation subtype 45

Property Editor
displaying 41

protocol data units. See PDUs

proxy authorization
ACI example 242
with cascading chaining 118

proxy DN 243

proxy right 197

PTA plug-in
configuring 445
examples 451
syntax 443
use in directory server 441

Q

quotation marks, in parameter values 54, 486

R

RC2 cipher 370

RC4 cipher 369, 371

read right 197

read-only mode 392
database 151

read-only replica 270
configuration 293

read-write replica 270
configuration 293

ref attribute 128

referential integrity
attributes 65
disabling 66
enabling 66
log file 64
modifying attributes 68
overview 64
using replication change log 66
with replication 65, 66

referral object class 128
referrals
  creating smart referrals 127
  creating suffix 129
  on update 79
  setting default 126
  suffix 78
renaming entries
  restrictions 58
replacing attribute values 59
replica
  exporting to LDIF 300
  read-only 270
  read-write 270
replica ID
  for a read-only replica 294
  for a read-write replica 293
replicate_now.sh script 302
replication
  and access control 250
  and password policy 262
  and referential integrity 65, 66
  and SSL 304
  cascading 285
  change log 271
  compatibility with earlier versions 272, 306
  configuration tips 290
  configuring a hub supplier 295
  configuring a read-only replica 293
  configuring a read-write replica 293
  configuring legacy replication 307
  configuring SSL 305
  configuring supplier settings 292
  consumer server 270
  consumer-initiated 270
  creating the supplier bind DN 291
  forcing synchronization 300
  hub supplier 270
  initializing consumers 297
  managing 269
  monitoring status 311
  multi-master 280
  of ACIs 250
  overview 270
  replica ID 293, 294
  replicate_now.sh script 302
  single-master 277
  solving conflicts 312
  supplier server 270
  supplier-initiated 270
  unit of 271
replication agreement 272
  creating 296
replication manager 271
required attributes
  creating 324
  deleting 324, 325
  editing 325
requiredObjectClass keyword 464
resource limits 266
  setting
    using command line 267
    using console 267
Resource Summary
  viewing 386
resource use
  connections 387
  monitoring 387
restoring data 144
  bak2db 148
  bak2db.pl 149
  dse.ldif 151
  from console 147
  replicated entries 150
restoring the database 413
retro change log
  and access control 311
  attributes 308
  object class 308
  searching 311
  trimming 310
retro change log plug-in
  enabling 309
  overview 273, 308
rights
  list of 197
roledn keyword 207
roles 156
  access control 165
  access to directory 207
  activating 265

attributes 163
editing 160
filtered
creating 159
example 164
inactivating 263
inactivation 161
managed
creating 158
example 163
nested
creating 160
example 165
object classes 163
overview 156
root DN, see Directory Manager 27
root DSE 490
root entry creation 480
root suffix
creating from command line 75
creating from console 74

S

SASL authentication 216
schema
checking 326
creating new attributes 320
creating new object classes 323
deleting attributes 321
deleting object classes 325
editing object classes 324
extending 317
nsslapd-schemacheck attribute 326
standard 317
viewing attributes 318
viewing object classes 322
schema checking
and access control 192
ldapmodify and 50
overview 326
turning on or off 326
search filters 493
Boolean operators 496
contained in file 491
examples 493, 496
matching rule 498
operators in 494
specifying attributes 494
syntax 493
using compound 496
using multiple 496
search right 197
search types, list of 494, 501
searches
approximate 495
equality 494, 497, 503
example 489
greater than or equal to 495, 503
international 497
international examples 502
less than 502
less than or equal to 495, 502
of directory tree 486
presence 495, 496
restricting scope of one-level 331
restricting scope of subtree 331
specifying scope 489
substring 494, 503
searching algorithm
overview 332
Secure Sockets Layer, see SSL 33
security
certificate-based authentication 371
LDAP URLs and 509
setting preferences 369
self access 203
LDIF example 204
self keyword 203
selfwrite right 197
example 240
server parameters
database
read-only 392
setting access controls 218
setting passwords 259
simple authentication 216
Simple Authentication and Security Layer (SASL).
See SASL authentication

Simple Network Management Protocol. See SNMP
Simple Sockets Layer. See SSL
single-master replication
    introduction 273
    setting up 277
smart referrals
    creating 127
    creating from command line 128
    creating from console 127
SNMP
    agents 400
    AIX SNMP daemon 406
    configuring 405
    managed device 399, 401
    managed objects 400
    master agent
        overview 400
        Unix 400
        Windows NT 400
    MIB
        entries table 404
        location of 402
        operations table 402
    monitoring the directory server 399
    NMS-initiated communication 401
    overview 399
    subagent
        configuring 408
        configuring contact 408
        configuring description 408
        configuring location 408
        configuring master host 408
        configuring master port 408
        configuring organization 408
        enabling 408
        overview 400
        starting and stopping on Unix 407
        starting and stopping on Windows NT 407
    traps 401
Solaris
    thread concurrency 387, 391
SSL
    and replication 304
    certificate password 33
    chaining with 104
    client authentication 373
    configuring clients to use 373
    enabling 368
    port number 31
    setting preferences 369
    starting the server with 33
SSL authentication 368
standard
    attributes 317, 318
    index files 331
    object classes 317, 322
standard schema 317
starting the directory server 29
    with SSL 33
start-slapd script 30
static groups 154
    creating 154
    modifying 155
stopping the directory server 29
stop-slapd script 30
sub suffix
    creating from command line 75
    creating from console 74
subagent
    configuring 408
    enabling 408
    overview 400
    starting and stopping on Unix 407
    starting and stopping on Windows NT 407
substring search 494
    international example 503
subtypes
    of attributes 44
suffix
    and associated database 71
    configuration attributes 76
    creating 38, 48
    creating from command line 75
    creating root suffix 74
    creating sub suffix 74
    custom distribution function 85
    custom distribution logic 85
    disabling 80
    in directory server 71
    using referrals 78
        on update only 79

with multiple databases 84
suffix referrals
creating 129
creating from command line 130
creating from console 129
supplier server 270
symbols
-, in change operation 55
::, in LDIF statements 474
<, in LDIF statements 473
“”, in ldapmodify commands 54
’’, in ldapsearch 486
syntax
ACI statements 188
attribute value 320, 321
LDAP URLs 505
ldapsearch 487
LDIF update statements 55
matching rule filter 498
search filter 493
system connections
monitoring 387
system indexes 331
system resources
monitoring 387

T

targattrfilters keyword 194
target
ACI syntax 188
attribute values 194
attributes 192
DNs containing commas 191, 242
keywords in ACIs 190
overview 189
using LDAP search filters 193
using LDAP URLs 203
target keyword 190
targetattr keyword 192
targetfilter keyword 193
targeting
directory entries 190
template entry. See CoS template entry.
thread
concurrency on Solaris 387
monitoring 387, 389
time format 512
timeofday keyword 214
traps 401
triple DES 370
Triple DES cipher 370, 371
tuning performance
database 410
server 409

U

unique attribute plug-in 455
configuring 461
creating an instance of 460
disabling 463
enabling 463
examples 465
markerObjectClass 464
requiredObjectClass 464
syntax 457
Unix
AIX SNMP daemon 406
master agent 400
user access 202
example 227
LDIF example 204
to child entries 203
to own entry 203
LDIF example 204
user and group management
referential integrity 64
user passwords 259
userattr keyword 208
restriction on add 212
user-defined attributes 318
user-defined object classes 322
userdn keyword 202
users
activating 265

inactivating 263
UTF-8 511

V

value-based ACI 194
viewing
    attributes 318

W

wildcard
    in LDAP URL 204
    in target 191
wildcards
    in international searches 500
    in matching rule filters 500
Windows NT
    master agent 400
write right 197